Tools for Working with Data

Nicole Sorhagen, Ph.D.

2020-06-26

Contents

Chapter 1

About this book

This book describes how to use R as a tool to work with data.

R statistics is becoming increasingly popular for data management and analysis due to its accessibility and versatility. For example, R can produce records of data analyses, which is consistent with the growing move towards reproducible and open science within the field of psychology. R statistics is also known for making elegant graphs, which can help develop data visualization skills. Because it is open-sourced it is extremely flexible - people create and share packages that make certain aspects of data analysis easy.

R is a programming language. Although learning a programming language can seem a bit intimidating, there are many benefits to trying to figure it out. Mastering the basics of R could be useful for your future coursework, as well as for data management and analysis needs outside the classroom (independent research, future employment, etc.). That is to say, Learning the basics of a programming language is a highly transferable skill.

The R programming language can be used within the R software as well as other programs. RStudio is a IDE (integrated development environment) and was designed to make the use of the R programming language more user friendly.

R, and its companion program RStudio, are free and available in PC, Mac, and Linux versions, so students can have it on their own computer - eliminating the need to visit computer labs or to buy student versions of expensive software. R can be downloaded from the CRAN (Comprehensive R Archive Network) (https://www.r-project.org/). Rstudio can be found here: https://rstudio.com/.

While you are welcome to download R and Rstudio on your personal computer, you do not have to for this course. We will be using Rstudio on a website called Rstudio cloud for class work (this is discussed in more detail in the next chapter). So I am not going to go into detail on downloading the programs on

to your computer here. Please email me if you are interested in this and are having a hard time figuring it out.

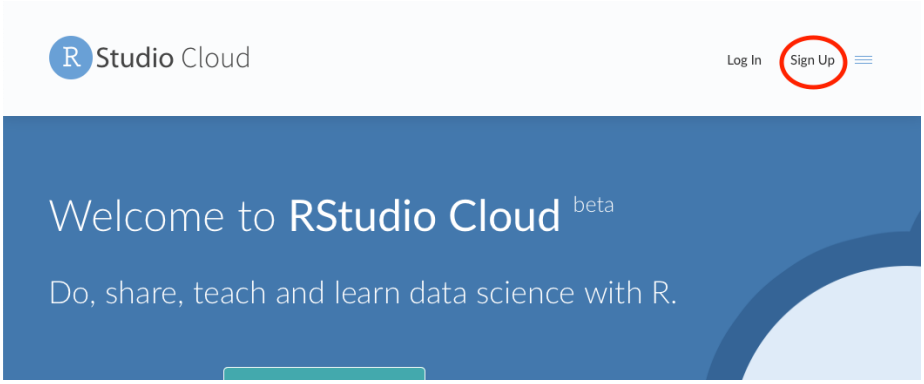
Finally, please note that I will be updating this book over the course of the semester.

Chapter 2

Set up project on Rstudio Cloud

We will use Rstudio cloud on this website: https://rstudio.cloud.

You must first make an Rstudio account by clicking the sign up button in the top right corner. (this is free)

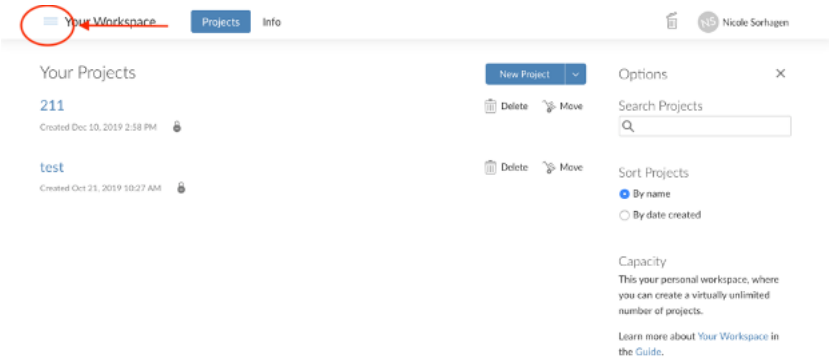


Then join our shared RStudio cloud workspace with the link that I sent you in the email titled 'Rstudio cloud shared workspace'.

You MUST join our shared workspace. I will be checking your work through this shared RStudio cloud workspace. Within this shared workspace, I will be able to see everyone's project, but you will only be able to see your project and my project.

Once you are in your Rstudio Cloud account...

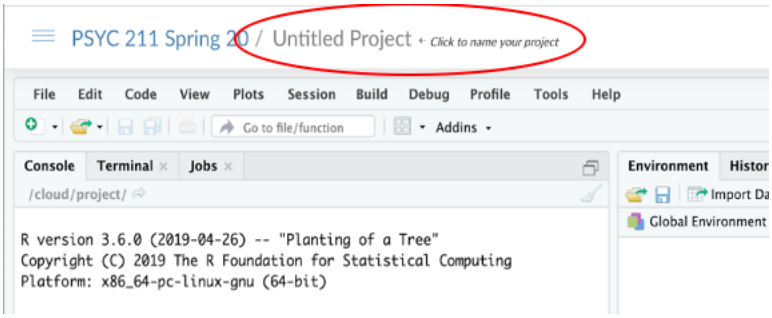
Expand the R studio cloud options by clicking on the 3 lines in the top left corner.



Then select our course (which will be titled the name of course and the semester). If you cannot see this option – then you have not been added to our shared workspace.

Once you are in the shared the classes workspace, open a new project.

Call this project your last name by clicking on the box that says 'Untitled Project' and typing your last name.



Chapter 3

Introduction

This chapter introduces the Rstudio cloud environment and describes how to import data into the RStudio cloud.

R cannot handle typos and is case sensitive ('Gender' is not the same as 'gender'). If your code will not run check for typos and caps. Related to this point, do not be afraid to copy and paste with using R. I often copy and paste code and replace variable or dataset names as needed. (This is one of the few times in education where copy and paste is OK!)

3.1 Layout of Rstudio cloud

Rstudio has four panes: the console panel, the script panel, the environment and history panel, and the files and plots panel. Each will be describe in turn next.

3.1.1 Console

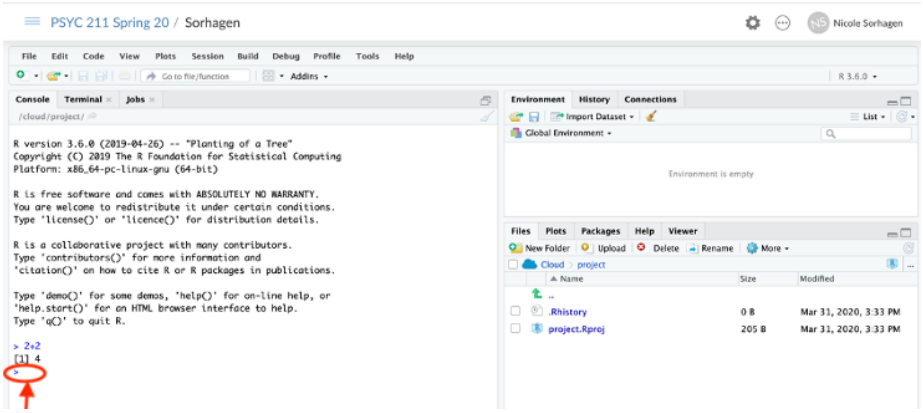
The console panel of R studio is where you can type commands and where you will see the output of commands.

In its most basic form, you can think of R as a fancy calculator.

For example:

In the console type `2+2` and then press RETURN on your keyboard. The answer '4' will apear on the next line.

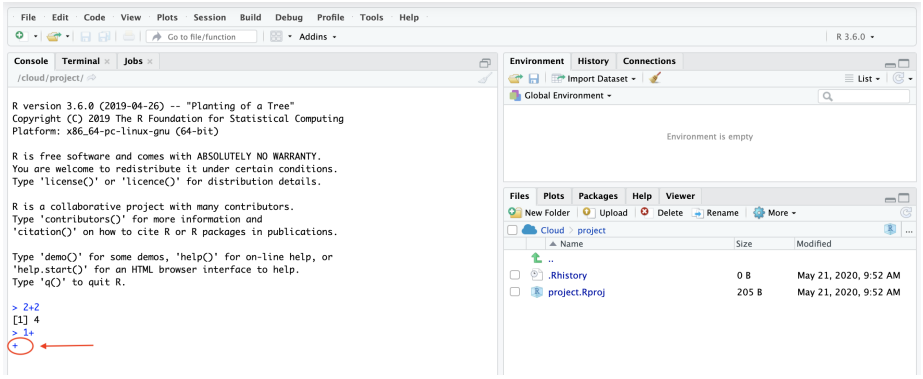
The `>` in the last line of the console means that the console is ready for a command (see red circle in the picture above).



If > is missing from the last line, it means that R is waiting for you to complete a command.

For example, type `1+` in the console and then hit enter.

The plus sign means the command is incomplete.



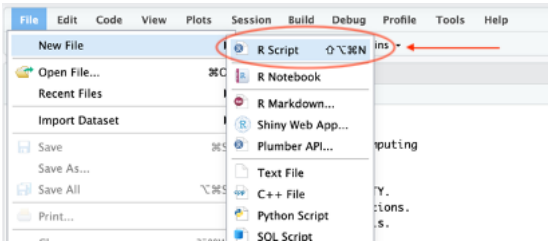
Push the **ESC** button on your keyboard to get back to the command prompt.

3.1.2 Script

One of the benefits of using R is that you can save a record of your work using scripts. Records of your work allow you to easily start and stop an assignment or research project. You can pick up where you left off whether it is 20 minutes later or 2 years later. It also lets you share with others – from professors, to collaborators, to peer reviewers.

To create a new script, go to the top bar menu:

FILE -> NEW FILE -> R SCRIPT



A new script will open in the top left of the RStudio platform.

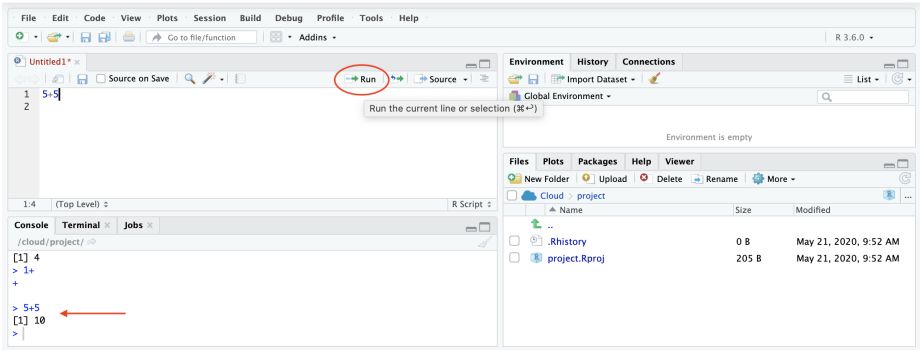
You should run code from scripts

Scripts are similar to running command in the console (this is what you did in the last section).

For example, type `5+5` in the script panel.

In order to run command in a script you should click the run button while the cursor is in the code or the code is selected. You can also run the code by pressing the COMMAND and RETURN keys on your keyboard at the same time (the ALT and RETURN key on a pc).

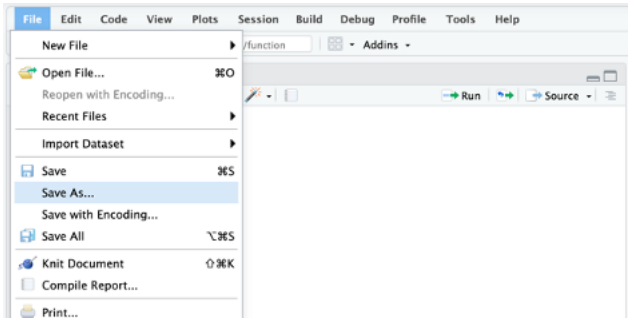
After the code is run, the results will automatically appear the in console (see red arrow in the picture below).



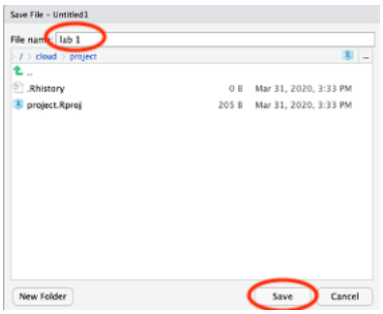
In order to use the script again you must **save** it.

From the drop-down menu select:

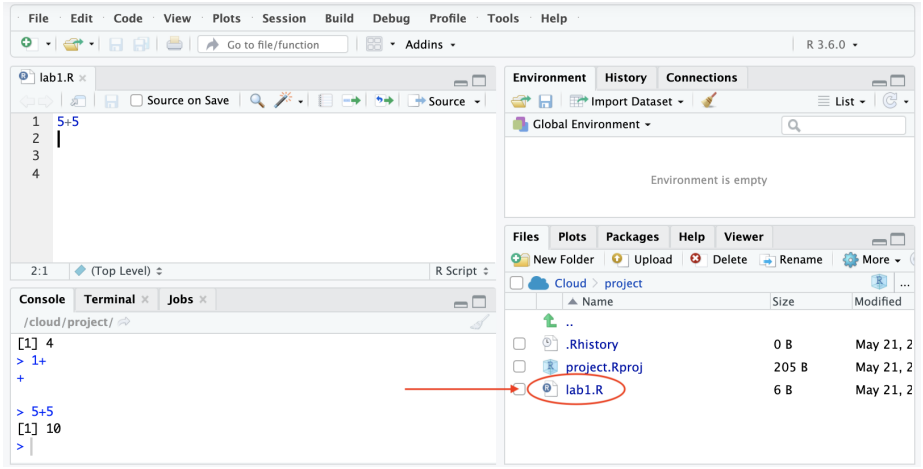
FILE -> SAVE AS



Type **lab 1** into the file name box. And then click the SAVE button.



Your file should now be listed in the files window in the bottom right.



This script file is a record of your work and is how you will be graded for this lab. Make sure you saved this file and complete the rest of this lab in your 'lab1' script.

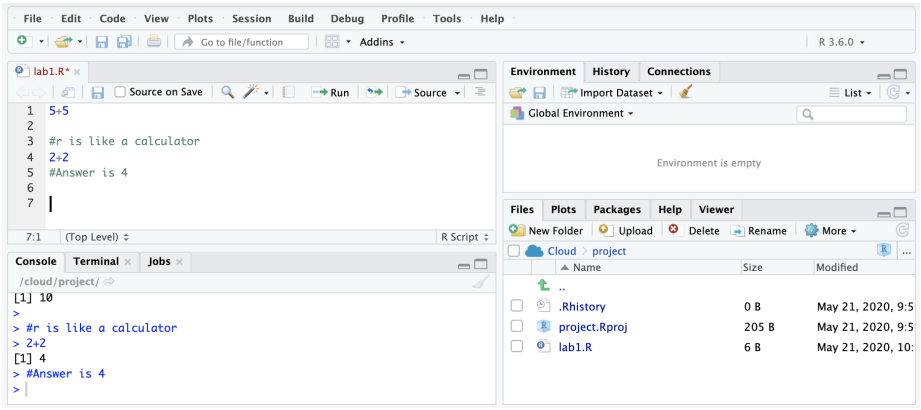
Within a script you should include comments to yourself and others using **#**. Anything with a **#** in front of it will not run. These comments and explanations are an important part of an R script.

For example, type the following in to the script and then run it.

```
#r is like a calculator

2+2

#Answer is 4
```



Note that the comments are green in the script.

3.1.3 Environment and history

In the top right corner of RStudio is the environment and history window. The **history tab** shows every line of code that has been run in the current session.

The **environment tab** is where all active **objects** are listed. An object is something can hold information for later use. The information can be data, values, output, or functions.

Objects are assigned using `<-`. Values on the right side of `<-` will be assigned to the object on the left side.

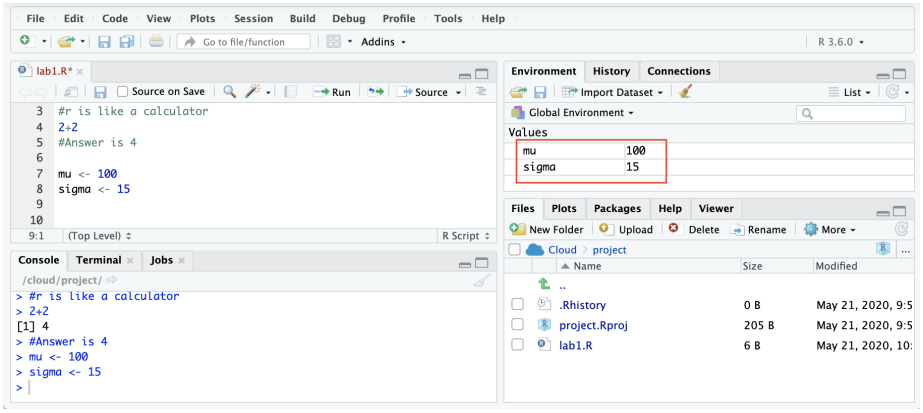
For example, let's tell R that the population mean of IQ scores is 100 and the population standard deviation is 15.

To do this use the following code:

```
mu <- 100

sigma <- 15
```

After you run these commands, the objects will now be listed in the environment panel in the top left.



The shortcut for making `<-` is the ALT and – key together. (or OPTION and – on a mac)

3.1.3.1 Vectors

It is possible to store more than one number in an object. One way to do this is to use a a **vector**. Assign a set of numbers a vector with the **combine** function: `c()`. Do use this, type all the numbers you want to store within the parentheses in a comma separated list.

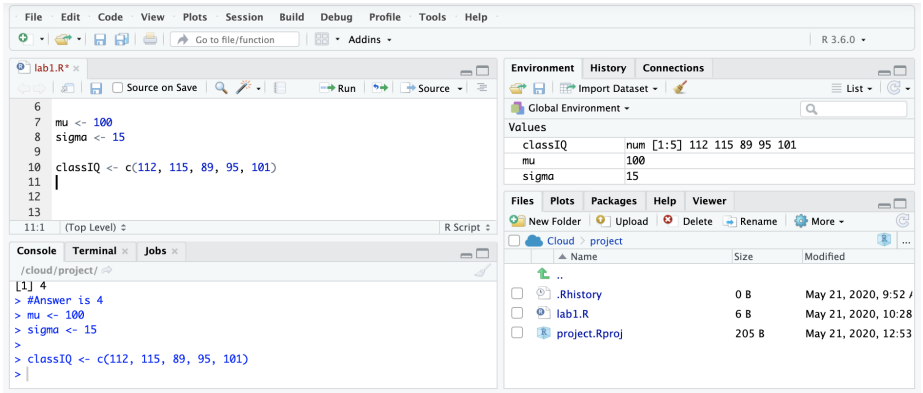
For example, let's enter IQ scores of students in a small class.

To do this use the following code:

```
classIQ <- c(112, 115, 89, 95, 101)
```

After you run this code,the classIQ vector should appear in the environment.

Here is a picture of what your screen should look like:



Calculations with vectors apply to all data points.

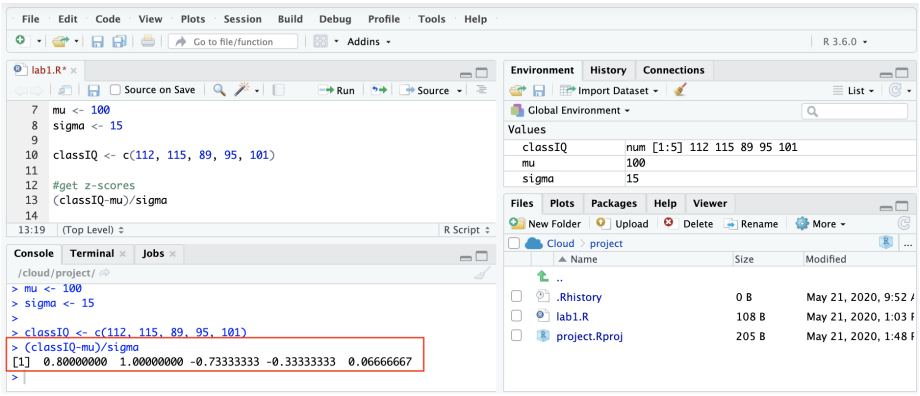
For example, let's calculate the z-scores for each of the IQ scores.

To do this use the following code:

```
#get z-scores

(classIQ-mu)/sigma
```

The results will appear in the console (See the red box in the picture below)



It is possible to save these answers as a vector using the `<-` function.

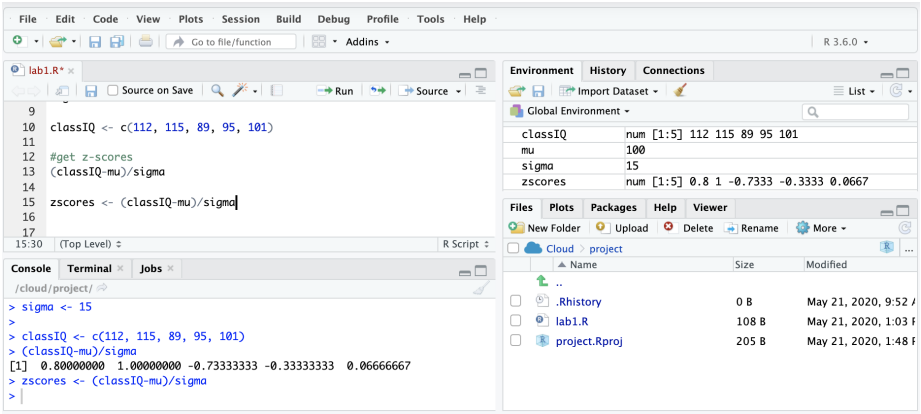
For example, let's save those zscores in a vector called zscores.

To do this use the following code:

```
zscores <- (classIQ-mu)/sigma
```

There should now be a vector in the environmnet called zscores.

Here is a picture so that you can check your progress:



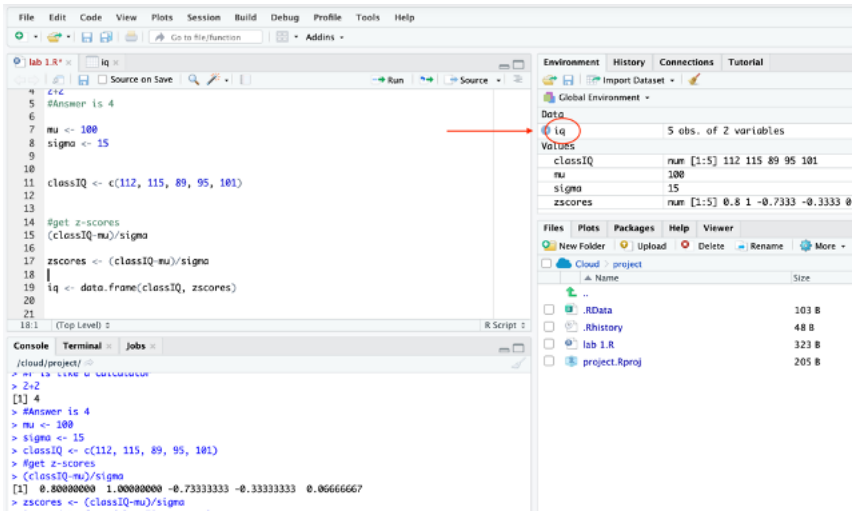
3.1.3.2 Data frames

Right now the IQ scores and the z-scores are in separate objects. Variables often need to be in a single object in order to do some basic analyses. You can combine the classIQ and the zscores variable using the **data.frame** command.

To do this use the following code:

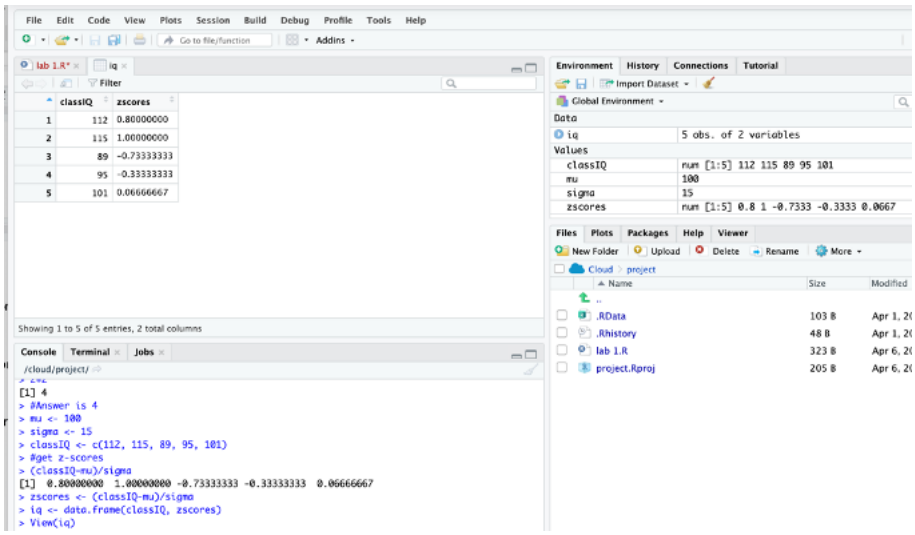
```
iq <- data.frame(classIQ, zscores)
```

This object will be listed under data instead of values in the environment panel.



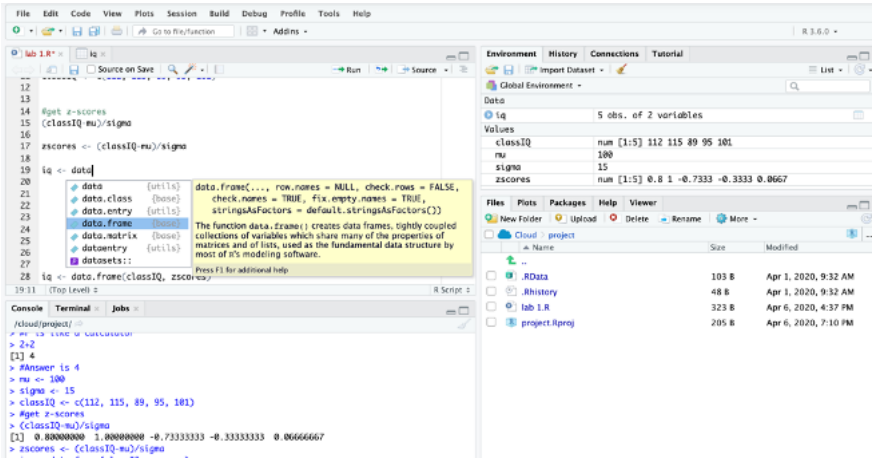
Double-click on the word 'iq' in the environment panel to look at the dataset that you just created (it is circled in red in the picture above).

A new tab will open with a spreadsheet view of the dataset. When you are done viewing the data, you can close it by click on the 'x' next to the name iq.



Note that after looking at the dataset this way, the command `view(iq)` appeared in the console. You can look at the dataset with the **view** command as well

Finally, when typing the code to create the data frame, you may have noticed that RStudio uses **predictive text**. This means that RStudio will suggest functions and objects as you type. You should take advantage of this nice feature!



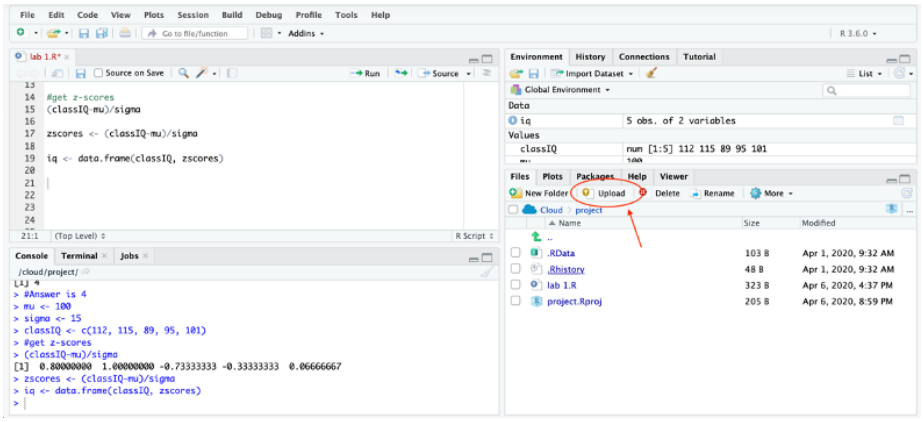
3.2 Importing data into Rstudio cloud

In the Introduction section you learned how to assign data to a **vector** using the **combine** function.

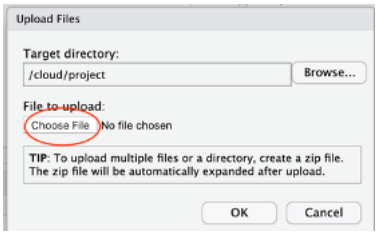
Another way to assign data to an object is by first entering the data into a spreadsheet (like google sheets or excel) and then import the data into RStudio. This will be our preferred method.

First download the exam2.csv file from d2l.

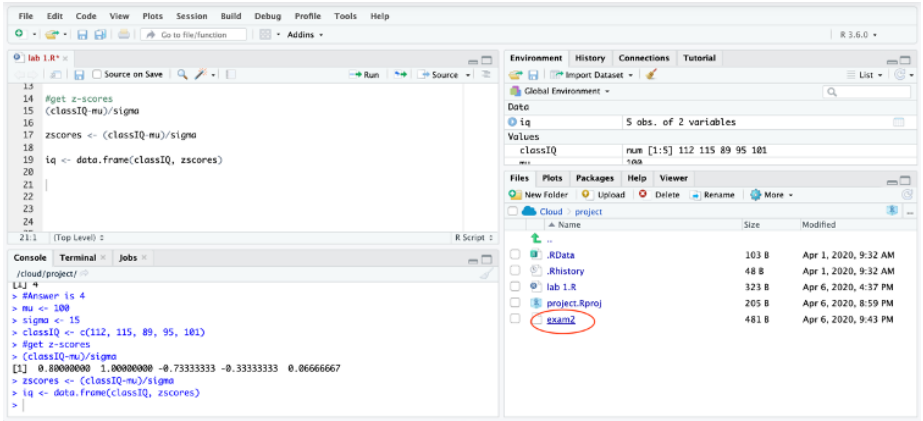
Then select the UPLOAD button in the files window.



In the Upload Files window, click the CHOOSE FILE button and then navigate to the exam2.csv file on your computer. Then click the OK button.

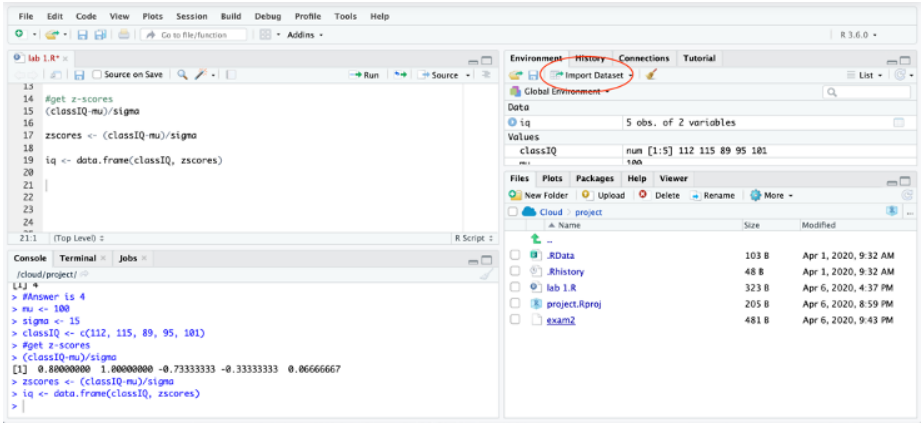


The data file should now be listed in the files section of RStudio.

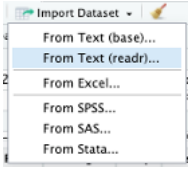


Then you need to import the data into the environment (i.e. assign the data to an object). This can be done through using point and click options or with code.

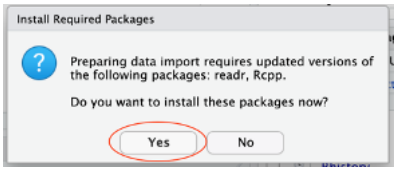
For point and click: First click on the **IMPORT DATASET** button in the environment panel.



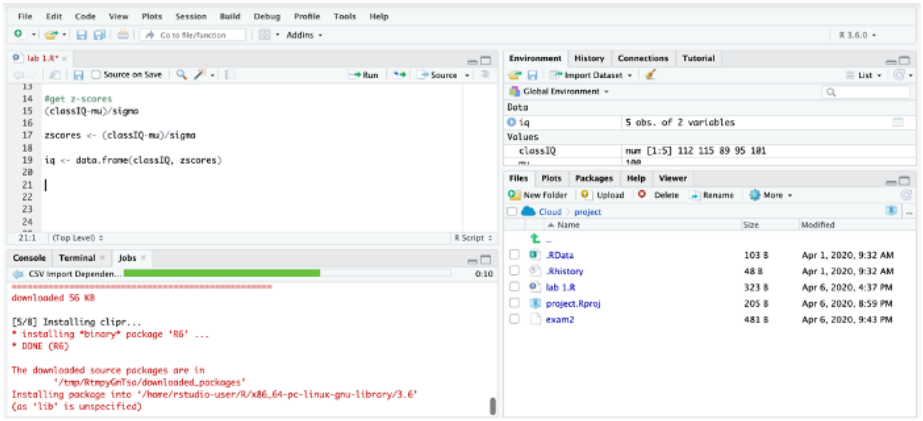
Then select the **'FROM TEXT (READR)'**



The first time you select this – the following window will appear asking if you would like to install the readr package. Select YES. I will introduce packages in the next section.

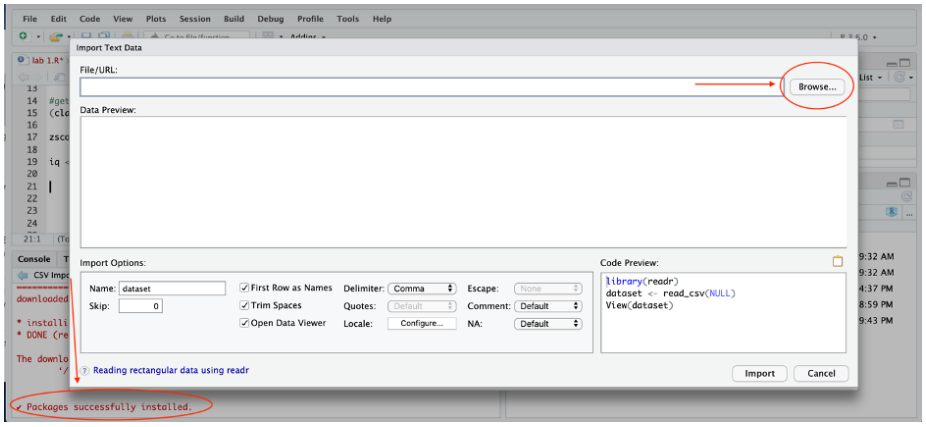


After you select yes, R will begin downloading the package. This can take a few minutes and will look something like this:



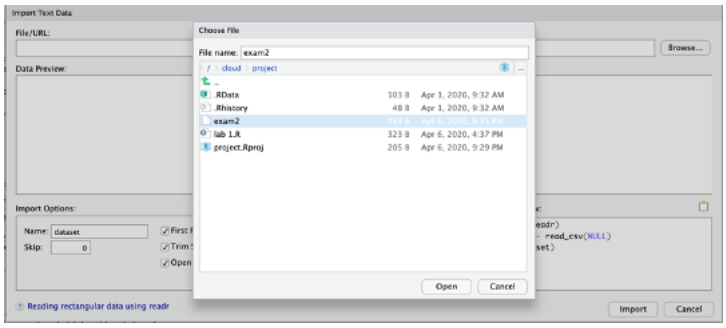
It is important to be *patient* here and let the package download completely before you move on to the next step.

When the download is complete, your screen should look like this:



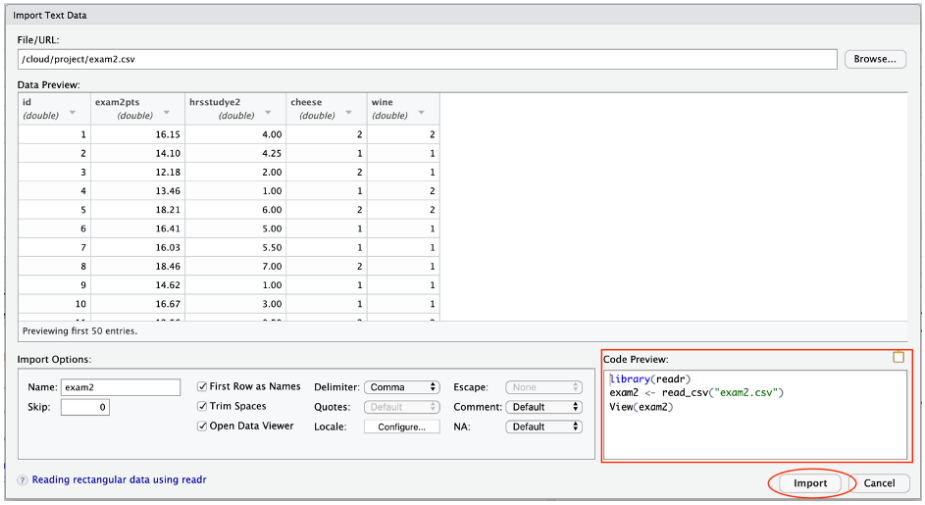
Note that you can see that the package was successfully installed in the console in the bottom left. The next time you use readr to import data – you will not have to download the package first.

Next select the BROWSE button in the top left corner of the import data window.



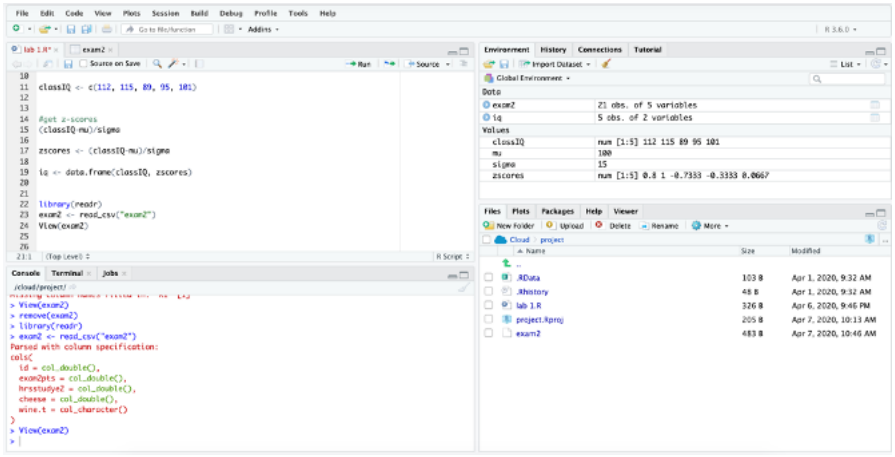
In the choose file window, select 'exam2'. And then select OPEN.

The next window should look like this:

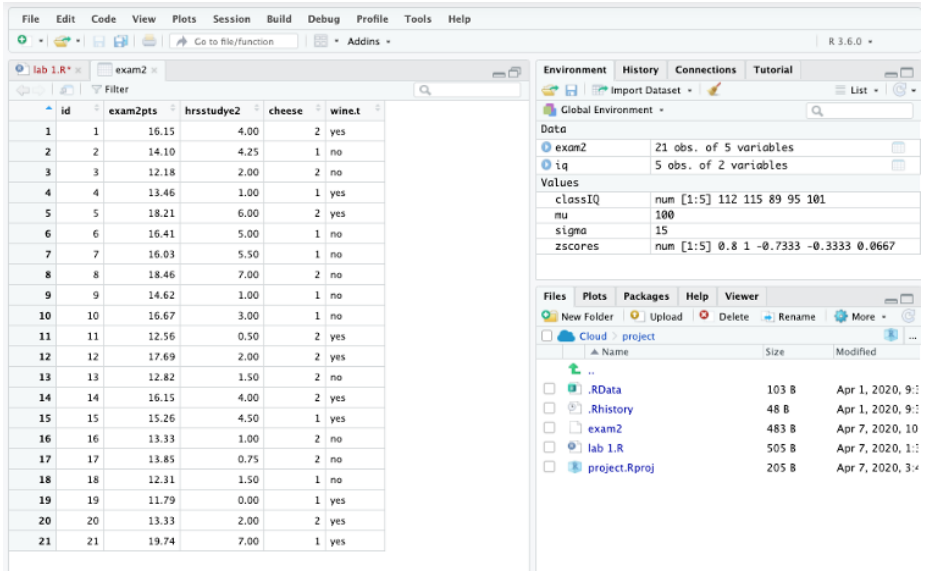


From here you should click the IMPORT button.

But first note the **Code Preview** box. This is the code you could use to import data (instead of clicking through all these windows). Copy this code before I click the import button and then paste it into your script for your records and in case you need to assign the file to an object again (because it is faster with code).



Double click on the word exam2 in the environment panel to look at the dataset.



	id	exam2pts	hrsstudye2	cheese	wine.t
1	1	16.15	4.00	2	yes
2	2	14.10	4.25	1	no
3	3	12.18	2.00	2	no
4	4	13.46	1.00	1	yes
5	5	18.21	6.00	2	yes
6	6	16.41	5.00	1	no
7	7	16.03	5.50	1	no
8	8	18.46	7.00	2	no
9	9	14.62	1.00	1	no
10	10	16.67	3.00	1	no
11	11	12.56	0.50	2	yes
12	12	17.69	2.00	2	yes
13	13	12.82	1.50	2	no
14	14	16.15	4.00	2	yes
15	15	15.26	4.50	1	yes
16	16	13.33	1.00	2	no
17	17	13.85	0.75	2	no
18	18	12.31	1.50	1	no
19	19	11.79	0.00	1	yes
20	20	13.33	2.00	2	yes
21	21	19.74	7.00	1	yes

Each column is a different variable. Each row is a different participant (in this example a student).
- The first column is an arbitrary student ID number – so that the students' identity is protected.
- The second column is exam points earned by the students out of 20 (this is real data from a Fall 2019 class).
- The third column is the number of hours the students studied for the exam (this is made up data).
- The fourth column is whether or not students ate cheese the night before the exam (1 = no; 2 = yes... also made up data).
- The last column is data on whether or not students drank wine the night

before the exam (1 = no; 2 = yes... also made up data).

Chapter 4

Packages

NOTE: Please open a new script and call it lab 2 (or week 2) for the replication of this chapter and the picturing data chapter assignment.

Base R refers to the functions that automatically come with R. But many people build on top of Base R to make R better. The way they do this is through **packages**, which contain new R functions. There are thousands of packages available that can do fancy things like quickly compute descriptive statistics and create APA style tables (and much much more).

The first time you use a package, you need to install it. Once a package is installed, you will need to tell R that you want to use it by loading it. You will need to load any packages you want to use each time you open the R program. (I am not exactly sure how this works in the RStudio cloud because it does not seem to shut down when you close out of the RStudio cloud website. See the Restarting R section below for a work around.)

That is, you only have to install a package once. You will have to load a package every time you want to use it.

4.1 Installing packages

The first time you use a package, you need to install it. We actually did this once already while importing data! This time let's learn more about the process.

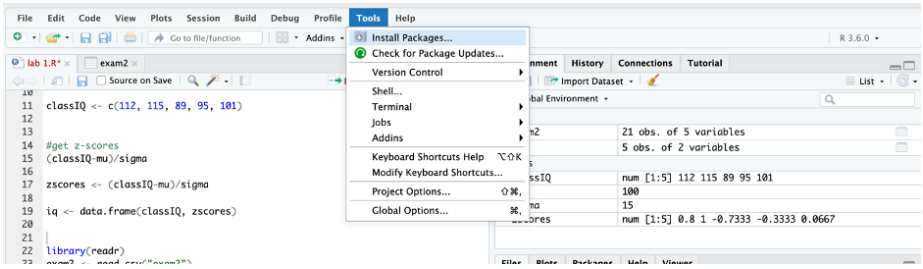
In RStudio, packages can be installed through point and click (GUI) or with code.

4.1.1 Installing packages using point and click (GUI)

Let's first install a package called **Tidyverse**. Tidyverse was created by Hadley Wickham and his team with the aim of making various aspects of data analysis in R easier. It is actually collection of packages that include a lot of functions (e.g., subsetting, transforming, visualizing) that many people think of as essential for data analysis. (See the tidyverse website for additional information: https://www.tidyverse.org).

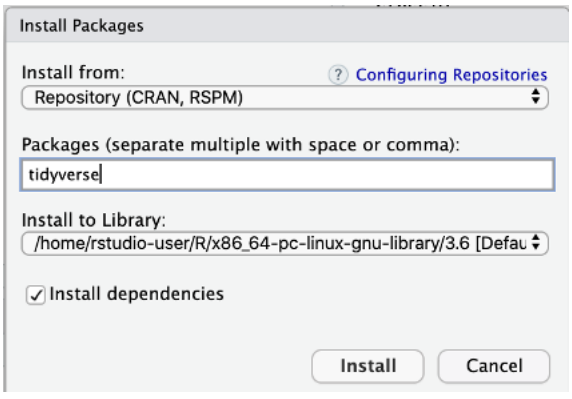
To install a package with GUI go to the top bar menu:

TOOLS -> INSTALL PACKAGES



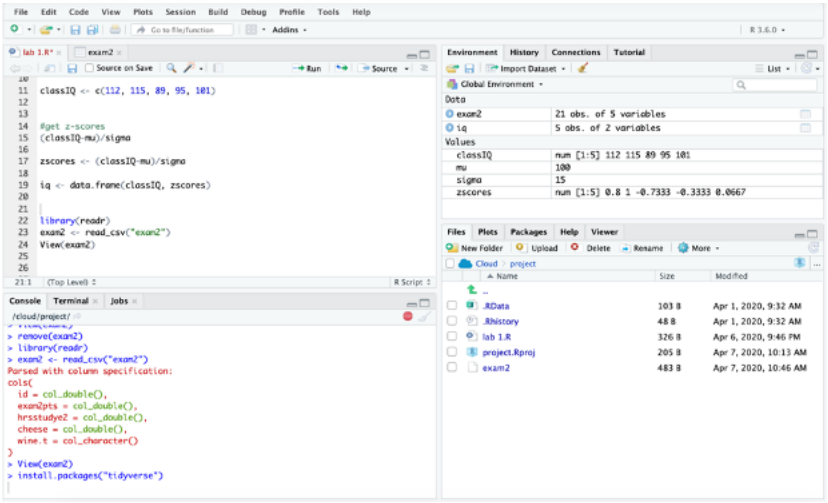
In the install packages window, type the name of the package you would like to install. For example, type `tidyverse` in the packages box.

Then click INSTALL.

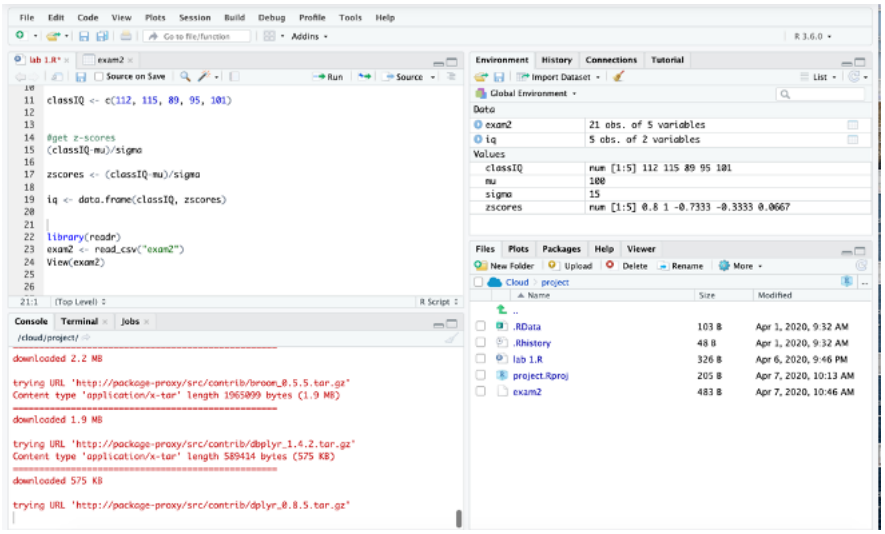


Again, installing a package can be a little slow on the RStudio cloud. Please be patient (maybe this is a good time to stretch your legs, refill your beverage, let the dog out, etc.)

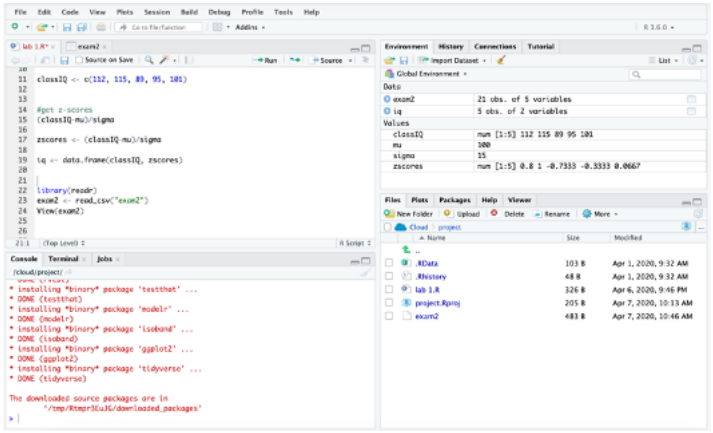
Your screen should look like this when it is starting to install:



It should look like this when it is in the process of installing:



And then this when the installation is complete:



Do not proceed until the console says the package has been installed.

4.1.2 Installing packages using code

You can also install a package using this code:
`install.packages()`

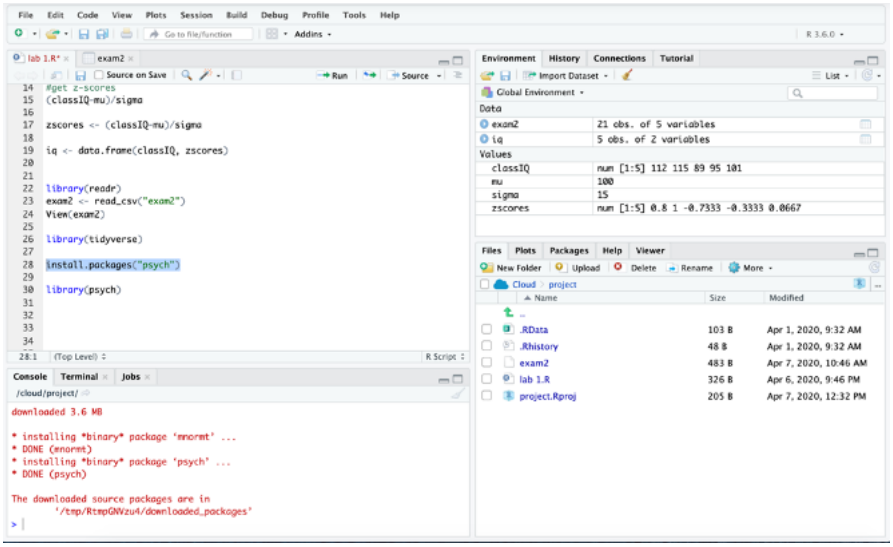
To install tidyverse, for example, you would use this code:
`install.packages("tidyverse")`
- Note that the word tidyverse is in quotes

But do not run this code – as you have already installed it with GUI.

Instead, let's install a package called psych using the `install.packages` command. The **psych package** is a package for personality, psychometric, and psychological research. It has been developed at Northwestern University (maintained by William Revelle) to include useful functions for personality and psychological research.

To install this package, use following command:
`install.packages("psych")`

Your screen should look like this when the package is completely installed:



Remember that installing packages is the first step to using them and they only have to be installed once.

Next let's learn how to load packages, so that you can use thier functions.

4.2 Loading Packages

Installing a package is only the first step.

In order to use a package, it must be loaded first.

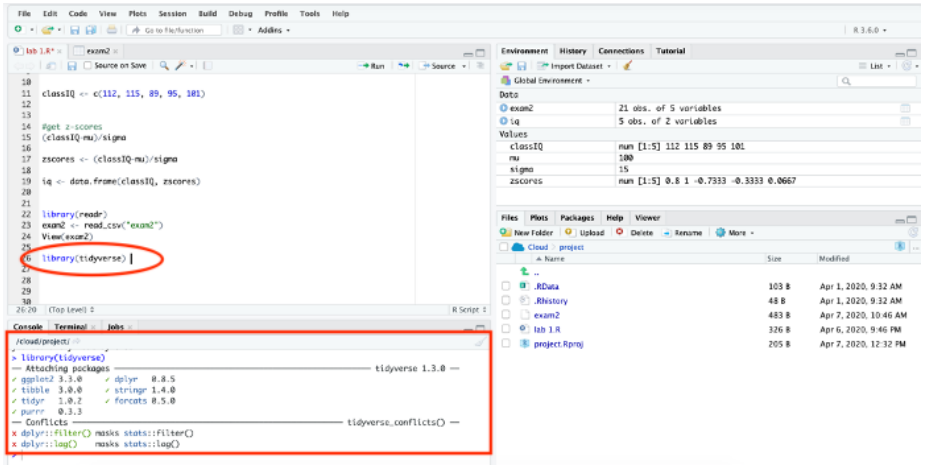
Packages can only be loaded with code. Packages need to be loaded every time you open the RStudio program. Most people's R scripts begin with the code that load packages.

When you have the Rstudio program installed on your computer this is straight forward (either the program is open or closed). This is less clear with Rstudio cloud because it does not seem to always shut down when you close the web browser site. (Please see the section on restarting Rstudio in the misc section below for a work around.)

The command to load a package is:
`library()`

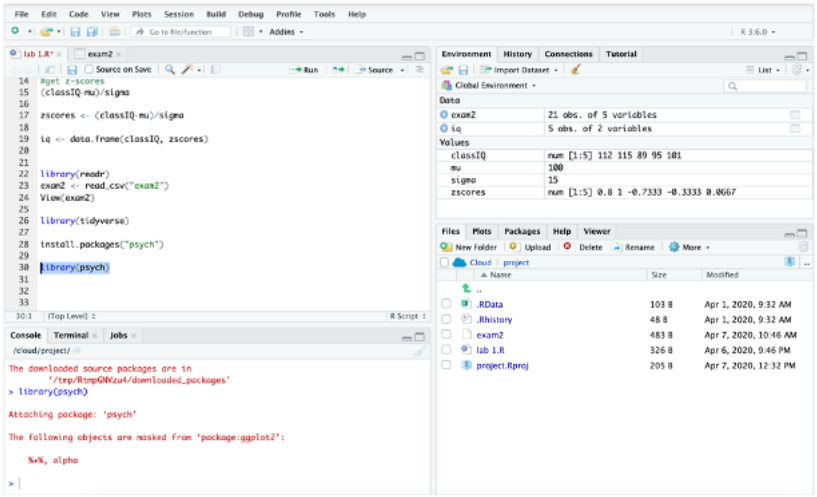
For example, load the tidyverse package with this:
`library(tidyverse)`

After you run this code, your screen should look like this:



The console shows that the Tidyverse package has been loaded (don't worry about the conflicts for now).

Next let's load the psych package using this command:
`library(psych)`



Again, don't worry about the warning about masked functions for now.

4.3 Misc

You can get additional information using the `help()` function and `?` help operator in R. They both provide access to documentation pages for all functions and packages.

For example, use the following code to get more information about Tidyverse:
`?tidyverse`

Or this command to get more information about Psych:

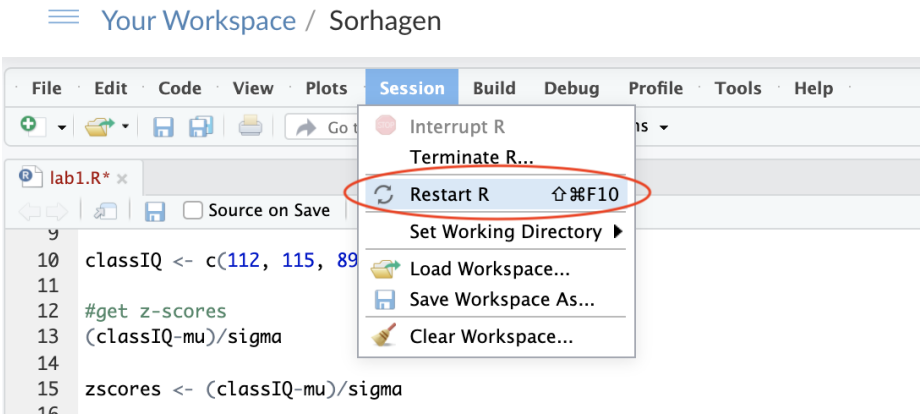
`help(psych)`

4.3.1 Restarting R

Because it is unclear whether Rstudio completely turns off when you close the website, you could restart the R session to simulate the act of closing and reopening the Rstudio program (like you could if it were installed on your computer).

To do this, in the drop down menu go to:

SESSION -> RESTART R



When you restart the R session, everything in the script, environment, console, and files will remain.

All packages that were loaded will be cleared, so you will have to reload them if you want to use them.

If something is not working like it is suppose to (and you have checked for typeos), try restarting the R session. It could be that the functions of one package conflict or mask the functions of another package.

Chapter 5

Picturing Data

"The simple graph has brought more information to the data analyst's mind than any other device." — John Tukey

This chapter focuses on how to make graphs and figures in R. Data visualization is useful for descriptive statistics, data analysis, and communicating results.

5.1 Histograms

Here you will learn how to make a histogram. Histograms plot the frequency of each score in a set of data. Thus, they are essentially a graphic of a frequency distribution. They are useful for checking the shape of a distribution (many statistical tests assume data is approximately normally distributed), checking for coding errors, and checking for outliers.

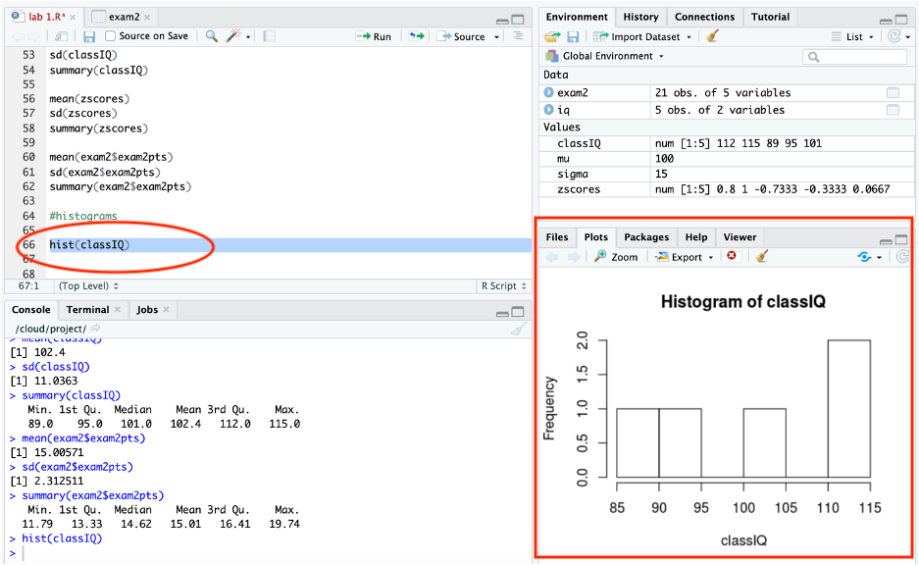
5.1.1 Histograms with base R

Let's first make histogram with base R by using the hist() function.

A **function** in R is any kind of operation. For example, the `hist()` function will create a histogram. An **argument** is what a function acts on.

For example, `hist(classIQ)` will return the histogram of the IQ scores in the classIQ vector. This code applies the function hist to the variable classIQ.

After you run this command, your screen should look similar to this:



I circled and boxed what should match here. (Please excuse a few differences between this screenshot and your screen, like the name of the script, the code in the script before the histogram, and the results in the console. I had presented the material in a different order the last time I taught it.)

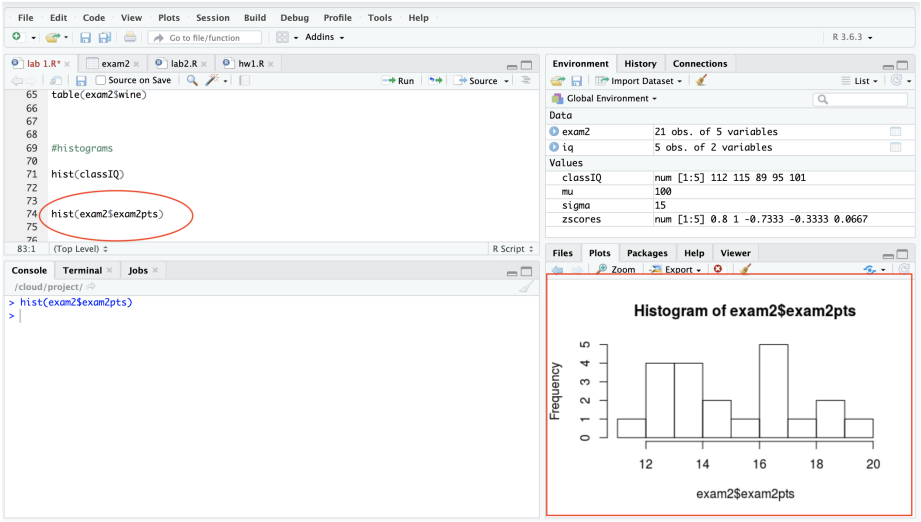
In order to use a base R function with a variable within a data frame you have to tell R to first look in the data frame in order to find the variable. You do this with the dollar sign (`$`). Place the `$` between the name of the data frame and the name of the variable.

For example, to use the `hist()` function to create a histogram of the exam 2 points variable in the exam 2 dataset, use this code:

```
hist(exam2$exam2pts)
```

- exam2$exam2pts is telling R to first go to the exam2 dataset and then use the exam2pts variable

Here is a picture:



The data looks some what normally distributed, with a slight positive shew.

5.1.2 Histograms with tidyverse

The base R option is quick and easy. But it is not customizable. Because of this – many people prefer to use **ggplot** (of the Tidyverse package - so tidyverse needs to be loaded).

Ggplot is typically taught with the analogy of a globe that is built one layer at a time. You start with a world of only ocean (no land). Then you progressively add "layers" of land, colors, terrain, legends, etc. This system is based on the grammar of graphics: statistical graphics map **data** onto perceivable **aesthetic attributes** (e.g., position, color, shape, size, line type) of **geometric objects** (e.g., points, bars, lines). Code can also be added to ggplots to make graphs in APA style.

With ggplot, you build plots step-by-step, layer-by-layer using the following steps:
1. Start with `ggplot()`
2. Supply a dataset and aesthetic mapping, `aes()`
3. Add on ...
+ **Layers**, like `geom_point()` or `geom_histogram()`
+ **Scales**, like `scale_colour_brewer()`
+ **Faceting Specifications**, like `facet_wrap()`
+ **Coordinate Systems**, like `coord_flip()`

The code for a histogram of the exam 2 points is:

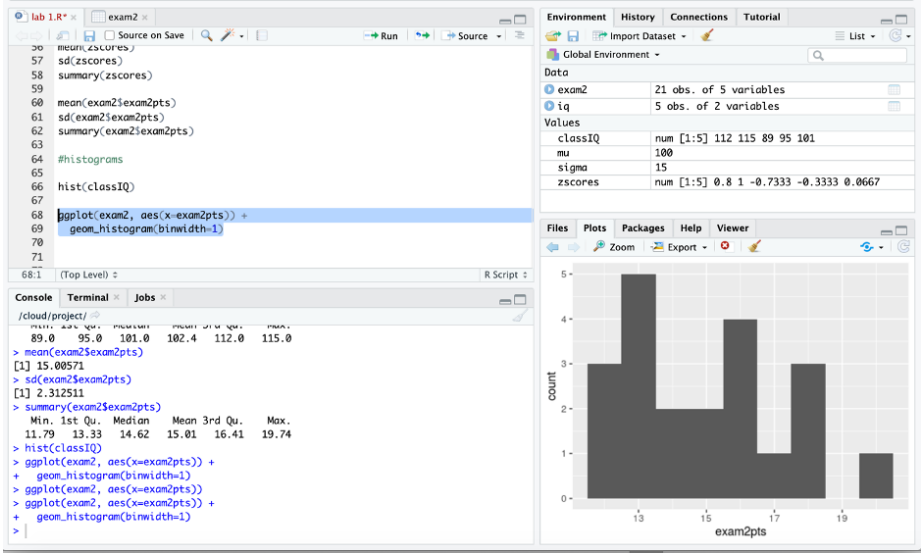
```
ggplot(exam2, aes(x=exam2pts)) +
geom_histogram(binwidth=1)
```

- The first line starts with ggplot and then supplies a dataset and aesthetic mapping, `aes()`

  - Because you supply the dataset this way - you do not need to use `$` to tell R where the variable is

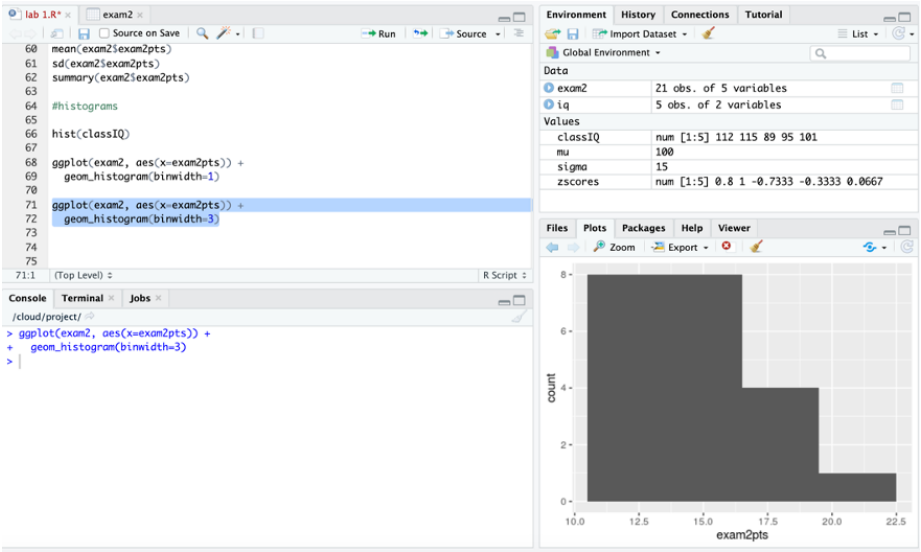
- The second line adds the layer of a histogram

After you run this code your screen should look like this:



Note that the histogram here is more detailed than the one you produced with base R. This is because base R used 5-points bins, while ggplot used 1-point bins (because you told R to). Here it is easier to see that the data is slightly skewed right.

In ggplot, it is easy to change the amount of points per bin by changing the number after the binwidth. For example, here I change the number to 3:



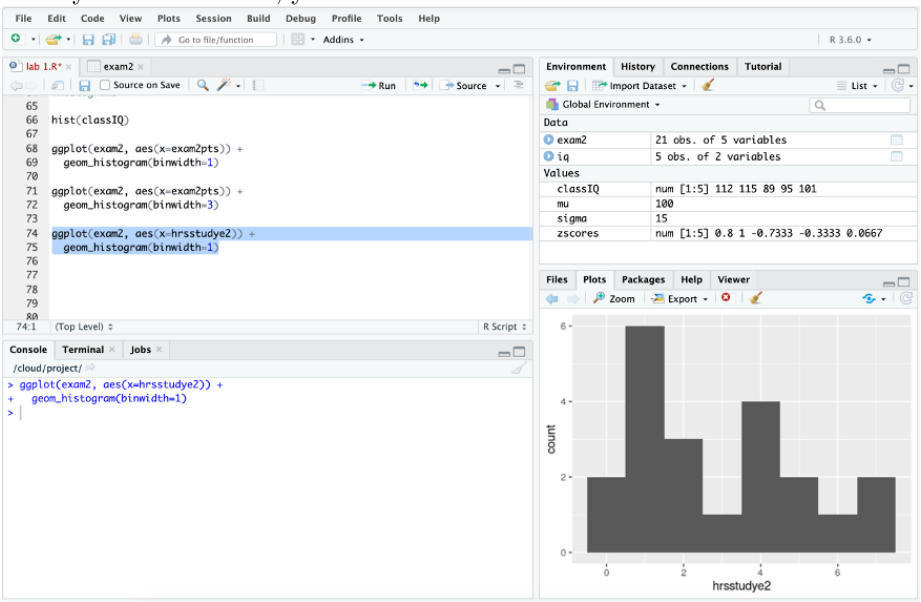
Some say that 10 bins in a histogram is a good rule of thumb (There are more precise equations for determining the "right" number of bins as well).

Let's look at a histogram of the number of hours studied for exam 2 next.

Here is the code:

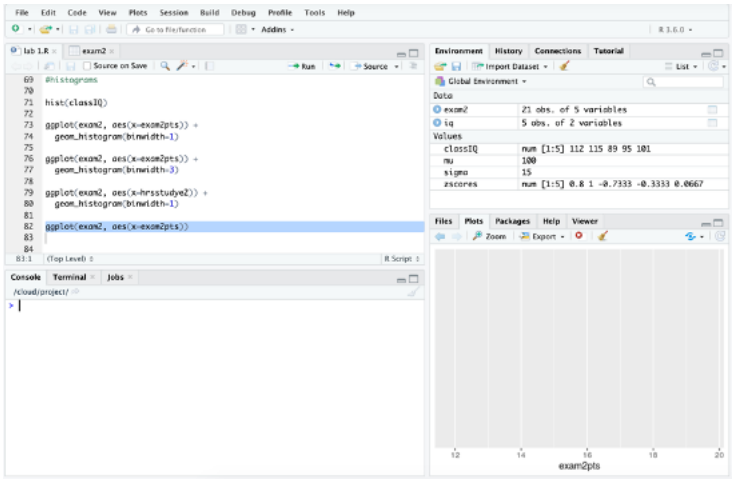
```
ggplot(exam2, aes(x=hrsstudye2)) +
geom_histogram(binwidth=1)
```

After you run this code, your screen should look like this:



The histogram show that data approximates the normal distribution and is roughly mound shape.

You do not need to run this – but I just want to show you that if I run only the first line of the histogram code, the figure would look like this:



. . . so this is the world as only ocean – without land. The second line of the ggplot code (i.e. `geom_histogram(binwidth=1)`) adds the "land".

Please note that I am going to start providing less screen shots of the whole Rstudio window from this point forward. When I include R code know that I mean that the code should be typed into a script.

5.2 Scatterplots

A **Scatterplot** is a graph where one variable is plotted on the y-axis and the other is plotted on the x-axis. Each dot represents one participant, measured on two variables.

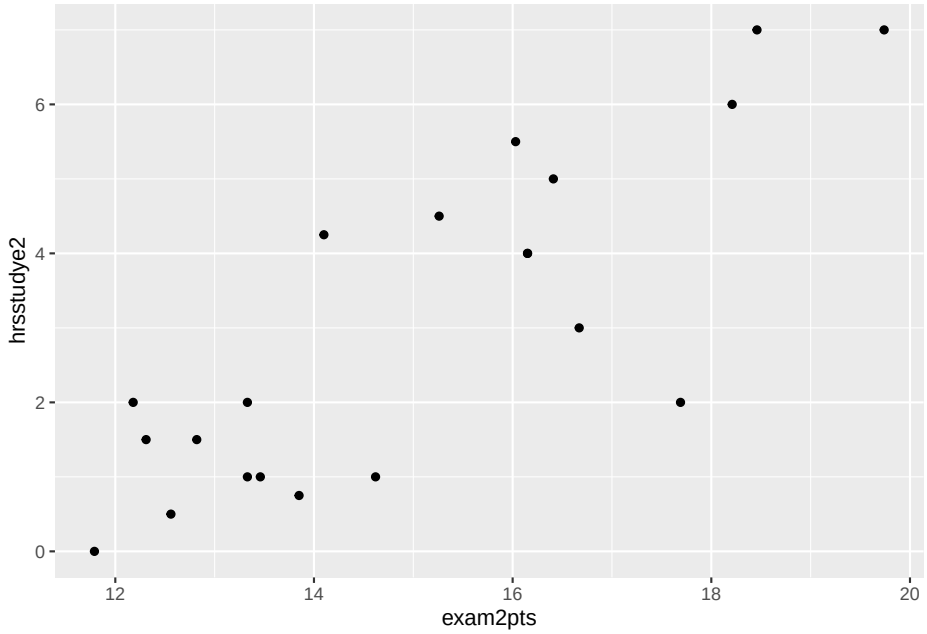
We are going to focus on using ggplots to create scatterplots because it is the more powerful data visualization tool in R.

5.2.1 Two continuous variables

Using the exam 2 dataset, let's say we hypothesized that there is a positive association between exam 2 scores and the number of hours studied for the exam. One of the first steps of exploring this association is to create a scatterplot.

Here is the code and resulting graph:

```
ggplot(exam2, aes(x=exam2pts, y=hrsstudye2)) +
  geom_point()
```



The data points are trending upward, suggesting a positive relation between exam 2 scores and the number of hours. The students who studied longer for the exam received higher grades; While those students who studied for less time received lower grades.

5.2.2 One continuous and one categorical variable

Let's say you were interested in the relation between cheese eating and exam 2 scores. You hypothesized that exam scores will be lowers for students who ate cheese the night before the exam because cheese gives nightmares.

Traditionally psychology likes to visualize the relation between a continuous and categorical variable using a bar graph. However, bar graphs can be misleading about the true nature of the data. Because of this, I prefer to continue to use a scatterplot to look at the association between a continuous and categorical variable - with some alterations to show the mean and variability (which is important to show with group data).

In ggplots you can alter the scatterplot to include the mean and variability, in addition to the actual data points, by including the `stat_summary()` function in the ggplot code. The `stat_summary()` function adds statistics to a ggplot.

To use the `stat_summary()` function you need to install the Hmisc package. It does not need to be loaded. *This is a rare exception on how packages in R normally work - The package does not need to be loaded in order for R to use it.*

Here is the code to install Hmisc:

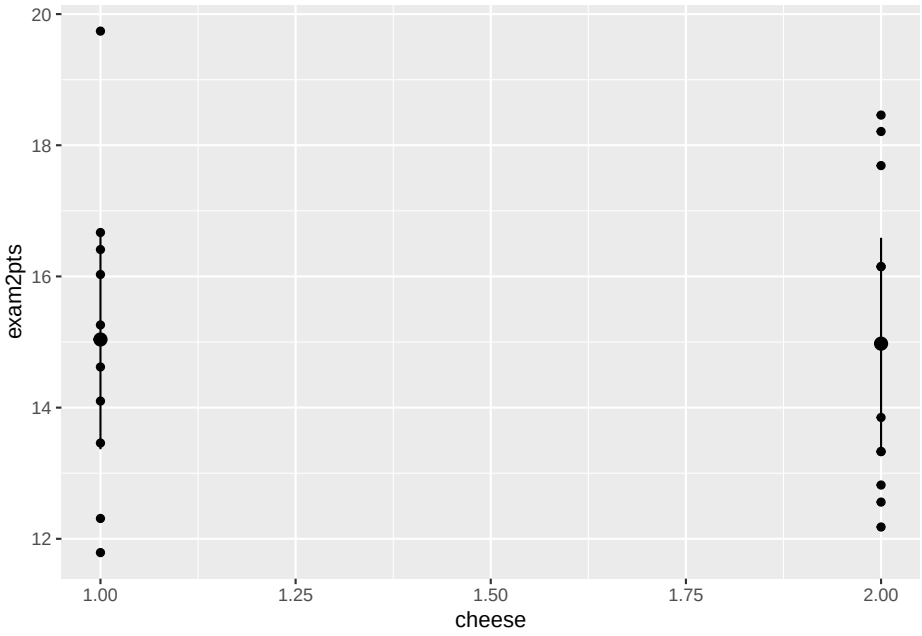
```
install.packages("Hmisc")
```
- remember to be patient and wait until the package is completely installed.

Then create the scatterplot with the following ggplot code:

```
ggplot(exam2, aes(x = cheese, y = exam2pts)) +
geom_point() +
stat_summary(fun.data = mean_cl_normal)
```

- x is the categorical variable
- y is the continuous variable
- `geom_point()` includes the data points
- The `fun.data = mean_cl_normal` within the `stat_summary()` function adds the mean and the confidence interval around the mean.

The graph should look like this:



The means are represented by the large dots. The lines represent the 95% confidence intervals, which shows the certainty around the mean and is based on the sample mean, standard deviation, and n.

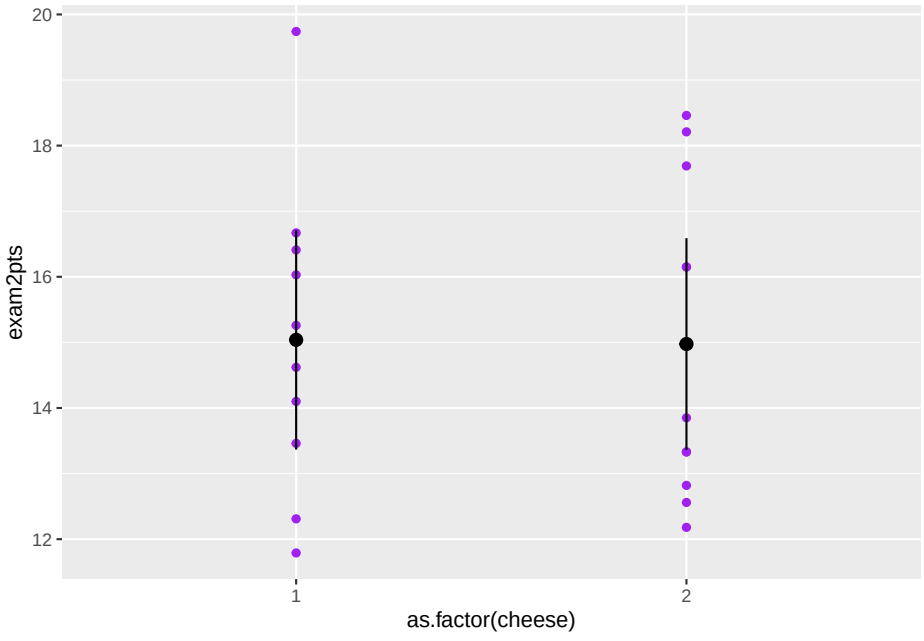
The small dots are the data points representing the participants cheese eating and exam grades.

Here you can see that the mean exam 2 scores are pretty similar for students who did and did not eat cheese the night before the exam. The spread of the scores is also similar. (Remember from the introduction chapter 1 = no and 2 = yes).

I like to make a few alterations to the previous code for aesthetics...

```
ggplot(exam2, aes(x = as.factor(cheese), y = exam2pts)) +
geom_point(color = "purple") +
stat_summary(fun.data = mean_cl_normal)
```

- The `color = "purple"` in the `geom_point()` function changes the color of the data points making the graph easier to read
- The `as.factor(cheese)` tells R to treat the cheese variable as a factor, which makes the x-axis more visually appealing



Chapter 6

Descriptive Statistics

NOTE: Please open a new script and save it as lab 3 (or week 3) for the replication of this chapter and the measurement chapter assignment.

This section focuses on functions that find descriptive statistics. **Descriptive statistics** refer to measures of central tendency (mean, median, and mode) and measures of variability (standard deviation, variance, range, etc.).

There are several functions that find descriptive statistics within R. My preferred method uses the Tidyverse and Psych packages, which I describe first. Next I will show you how to find descriptive statistics using base R.

6.1 Descriptive statistics using Tidyverse and Psych packages.

First load the tidyverse and psych packages (if they are not already loaded)

```
library(tidyverse)
library(psych)
```

The `describe()` function of the Psych package was made to produce the most frequently requested stats in psychology research in an easy to read data frame. I pair this with tidyverse styled code (because of the piping - which I explain next).

Here is the code to get descriptive statistics for the exam2 dataset we made:

```
exam2 %>%
describe()
```

- The `describe()` function applies to all of the variables in the dataset (here exam2)
- The `%>%` in this code is called a **pipe**
- Pipes are part of the Tidyverse package
- The shortcut to write a pipe (`%>%`) is `shift + command + M (shift + alt + M on a pc)`
- Pipes are a way to write strings of functions more easily
- You can think of it as a "THEN"
- So this code would be read as "use the exam2 dataset" THEN "compute descriptive statistics with the describe function"

The twitter handle WeAreRladies uses this example to show the sequential nature of a pipe (%>%):
I woke up %>% showered %>% dressed %>% glammed up %>% took breakfast %>% showed up to work

Let's look at the results

```
##            vars  n  mean   sd median trimmed  mad   min   max range  skew
## id            1 21 11.00 6.20  11.00   11.00 7.41  1.00 21.00 20.00  0.00
## exam2pts      2 21 15.01 2.31  14.62   14.88 2.65 11.79 19.74  7.95  0.37
## hrsstudye2    3 21  3.02 2.20   2.00    2.88 2.22  0.00  7.00  7.00  0.41
## cheese        4 21  1.52 0.51   2.00    1.53 0.00  1.00  2.00  1.00 -0.09
## wine          5 21  1.48 0.51   1.00    1.47 0.00  1.00  2.00  1.00  0.09
##            kurtosis   se
## id            -1.37 1.35
## exam2pts      -1.13 0.50
## hrsstudye2    -1.26 0.48
## cheese        -2.08 0.11
## wine          -2.08 0.11
```

The results show that the average exam points was 15.01 (out of 20), with a standard deviation of 2.31 points. Students studied for the exam for an average of 3.02 hours (SD = 2.20).

As the cheese and wine variables are nominal, the mean and standard deviation are not particularly meaningful. Also, any statistics on the ID numbers are meaningless.

This is a good place to mention that it is vital that you as a researcher understand what the numbers you are looking at are and the assumptions that they carry. R (or any computer program) will not tell you if what you asked for does not make sense or is not appropriate.

Rather than getting meaningless results that you have to ignore, you could add the `select()` function to the command above to select certain variables within

a dataset. Note that you must include more than one variable for the `select()` function. Use the `pull()` function if you want to select only one variable.

For example, to select the exam2pts and hrsstudye2 variables use the following code:

```
exam2 %>%
  select(exam2pts, hrsstudye2) %>%
  describe()
```

```
##            vars  n  mean   sd median trimmed  mad   min   max range skew
## exam2pts      1 21 15.01 2.31  14.62   14.88 2.65 11.79 19.74  7.95 0.37
## hrsstudye2    2 21  3.02 2.20   2.00    2.88 2.22  0.00  7.00  7.00 0.41
##            kurtosis   se
## exam2pts      -1.13 0.50
## hrsstudye2    -1.26 0.48
```

You should create frequency tables for nominal data. Do this with the `count()` function.

For example, create a frequency table for the cheese variable with this code:

```
exam2 %>%
  count(cheese)
```

```
## # A tibble: 2 x 2
##   cheese     n
##    <dbl> <int>
## 1      1    10
## 2      2    11
```

- this code is saying to "use the exam 2 dataset and then count the cheese varible.

The results show that 10 students did not eat cheese the night before Exam 2 (see Introduction for codebook – or what the 1 and 2 mean) and 11 students did eat cheese the night before the exam.

Finally, often we want to know descriptive statistics by group. For example, say you were interested in relation between cheese eating and exam 2 scores. You would want to know the descriptive statistics of the exam 2 scores for the students who did and did not eat cheese.

To do this I use the `describeBy()` function of the psych package, which reports basic summary statistics by a grouping variable. You have to tell R where to find the grouping variable by first including the dataset, followed by a `$` and the

variable name. In this example: `exam2$cheese` (I can't figure out how to avoid the $ here - I will give extra credit if you can.)

Use the `pull()` function to select the exam2pts variable.

```
exam2 %>%
  pull(exam2pts) %>%
  describeBy(exam2$cheese)
```

```
##
##  Descriptive statistics by group
## group: 1
##    vars  n  mean   sd median trimmed  mad   min   max range skew kurtosis   se
## X1    1 10 15.04 2.34  14.94   14.86 2.19 11.79 19.74  7.95 0.41    -0.75 0.74
## ------------------------------------------------------------
## group: 2
##    vars  n  mean  sd median trimmed  mad   min   max range skew kurtosis   se
## X1    1 11 14.98 2.4  13.85    14.9 2.48 12.18 18.46  6.28 0.28    -1.78 0.72
```

The results show that the average exam 2 score for students who ate cheese was 15.05 (SD = 2.34) and the average exam 2 score for students who did not ate cheese was 14.98 (SD = 2.4). Other statistics that you might find useful are the group's n, median, minimum and maximum scores, range, and standard error (se).

6.2 Descriptive statistics using base R

Descriptive statistics can also be computed using base R.

When a variable is stored directly in an object, you can apply the mean and standard deviation functions to the object.

For example:

```
mean(classIQ)
```

```
## [1] 102.4
```

```
sd(classIQ)
```

```
## [1] 11.0363
```

The summary function provides the range and median as well:

```
summary(classIQ)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    89.0    95.0   101.0   102.4   112.0   115.0
```

Remember that if a variable is in a data frame, you have to tell R to first look in the data frame in order to find the variable. You do this with the dollar sign ($). Place the $ between the name of the data frame and the name of the variable.

For example, to find the average points earned on Exam 2 use the following code:

```
mean(exam2$exam2pts)
```

```
## [1] 15.00571
```

Note that when typing this code RStudio will provide a list of the variables in the exam2 after you type the $. It is very convenient.

The `table()` function of base R performs categorical tabulations of data, frequency tables, and cross tabulations.

For the present example, the code is:

```
table(exam2$cheese)
```

```
##
##  1  2
## 10 11
```

Note that the table is laid out differently than the Tidyverse one above. But you can still easily see that 10 students did not eat cheese the night before Exam 2 and 11 students did eat cheese the night before the exam.

Some people think that the Tidyverse and Psych packages make computing descriptive statistics a bit easier/more direct/better/easier to understand than base R. You should decide which you prefer. (I tend to prefer Tidyverse and Psych). Another packages that compute descriptive statistics is skimr

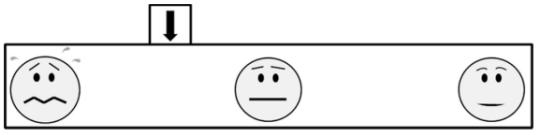
Chapter 7

Measurement

Math anxiety is the feeling of tension or worry in situations that involve math and is a major predictor of math achievement and career choices (Foley et al., 2017; Hembree, 1990).
Ramirez and colleagues (2013) developed the Child Math Anxiety Questionnaire (CMAQ) to measure math anxiety of young children. The measure (which can be seen below) consists of 8 questions which children responded to on a sliding scale that ranged from 1 to 16 points (the points were invisible to the children). Ramirez and colleagues (2013) calculated each child's CMAQ score by computing an average score of the eight items. Low scores on the CMAQ indicates high levels of math anxiety.

Child Math Anxiety Questionnaire Items

1. How do you feel when taking a big test in your math class?
2. How would you feel if you were given this problem? *There are 13 ducks in the water. There are 6 ducks in the grass. How many ducks are there in all?*
3. How would you feel if you were given this problem? *You scored 15 points. Your friend scored 8 points. How many more points did you score than your friend?*
4. How do you feel when getting your math book and seeing all the numbers in it?
5. How do you feel when you have to solve $27 + 15$?
6. How do you feel when figuring out if you have enough money to buy a candy bar and a soft drink?
7. How do you feel when you have to solve $34 - 17$?
8. How do you feel when you get called on by the teacher to explain a math problem on the board?



Let's pretend that you completed a pilot study testing the construct validity (if it is a good measure of math anxiety) of the CMAQ before Ramirez and her colleagues studied its relation to working memory and math achievement.

You gave the CMAQ to a convenience sample of 40 second graders. In addition to measuring the students' math anxiety with the CMAQ, you also measured the students' math ability, general anxiety, and you gave them a different measure of math anxiety.

Here is the data.

The first column is arbitrary ID numbers to identify the participants. The next 8 columns represent your participants responses on the CMAQ items. You will see that column J, titled cmaq, is blank. We will complete this column next in the brief introduction to spreadsheets section.

The next column (genanx) are a measure of general anxiety which was operationalized as the participants scores on the short form of the State - Trait Anxiety Inventory (STAI) which includes 6 statements - rated on a 1 to 4 point scale. The range is 6 to 24 points, with 6 points signifying no anxiety and 24 points signifying the highest level of anxiety.

The sema column is participants scores on the Scale for Early Mathematics Anxiety (SEMA), which is another measure of children's math anxiety (Wu et al., 2012). This measure consists of 20 items rated on a 4 point scale (0 - not nervous at all to 3 - very nervous). Responses are summed. Higher scores on the SEMA indicates high levels of math anxiety.

The wjap column is the participants w-scores on the applied problem subscale of the Woodcock-Johnson III, which consists of math related word problems. The W-scores are Rasch transformed and centered on 500.

t2cmaq is the participants' CMAQ scores administered 2 weeks later.

7.0.1 A brief introduction to spreadsheets (and some brief review of previous material)

Researchers often initially enter data into spreadsheets (excel, google sheet, numbers, etc.). So, I would like to briefly review how to use basic functions in a spread sheet. *We will learn how to create a new variable using R in the data transformation chapter.*

Either save a copy of the measurementma data to your own google account, or download the data and open it in excel or numbers (or whatever spreadsheet you prefer).

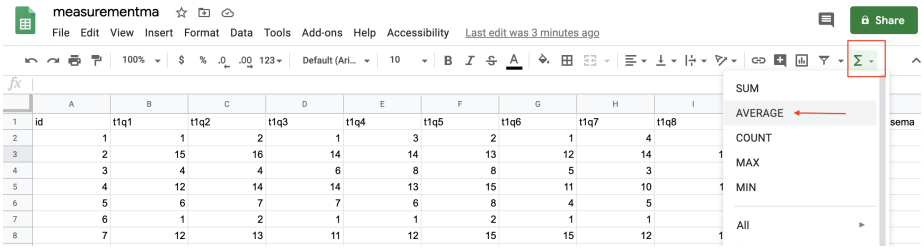
You may have noticed that there is no data in the cmaq column. To complete this column, we need to calculate the average responses of the CMAQ items for each participant. One way to do this is to use the function options within the

spreadsheet. (You also could enter the mean formula using the = sign, which I am not going to show here. Let me know if you would like me to post a short video showing how to do this.)

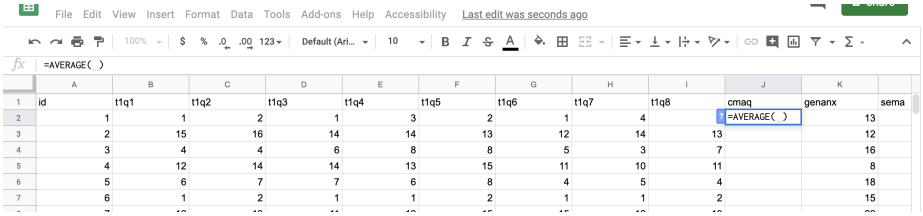
My screenshots show how to use the spreadsheet function options in google sheets. However, the process is very similar across all spreadsheet programs - you just might have to look in a different place on your screen.

First click in the first cell in the cmaq column (cell J2).

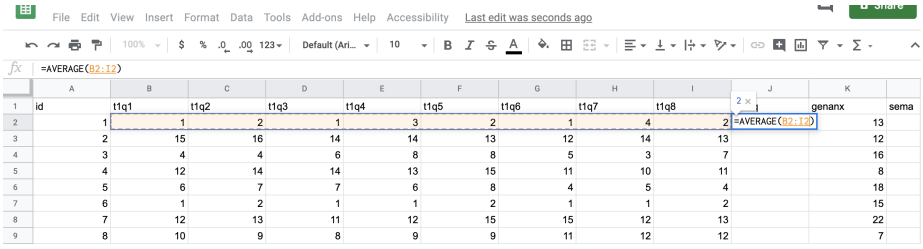
Then click on the sum symbol in the far right of the icon menu, and select the average option.



Then you should have the average function in cell J2 with nothing in the parenthesis.

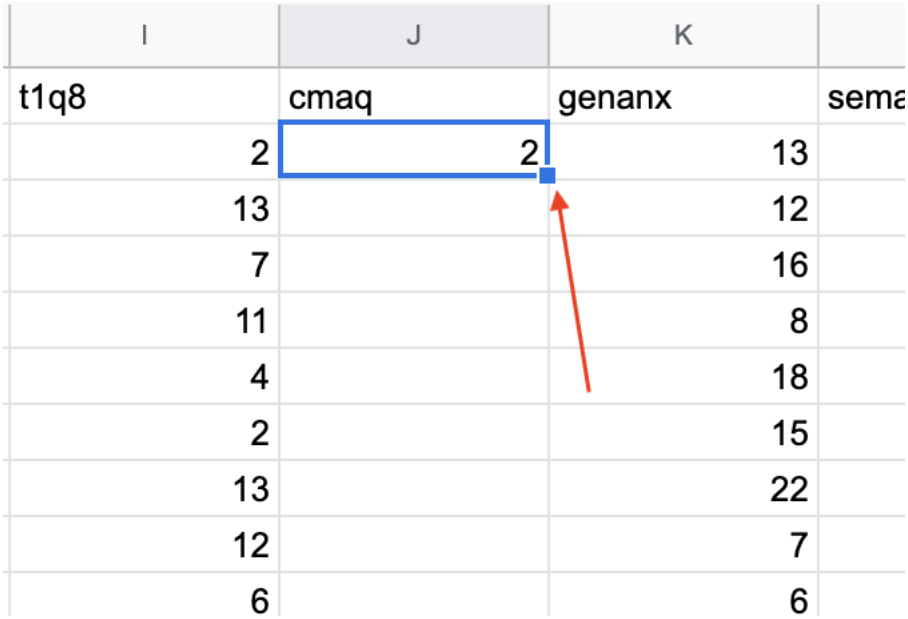


Next you need to tell the spreadsheet which cells you want it to average by listing them in the parenthesis. You can either select the cells you want the spreadsheet to average or you can type the names of the first and last cell separated by a : sign (in this example B2 : I2). Then hit enter on your keyboard and the average of the 8 items will appear (2).



You do not need to enter the equation separately into each cell of this column because spreadsheets will autofill equations for you. To do this, select the cell containing the formula, then select the small square in the bottom right corner

of the cell and drag it down to the last row in the dataset. (In some programs you can double-click on the small square and it will autofill to the bottom of the column).



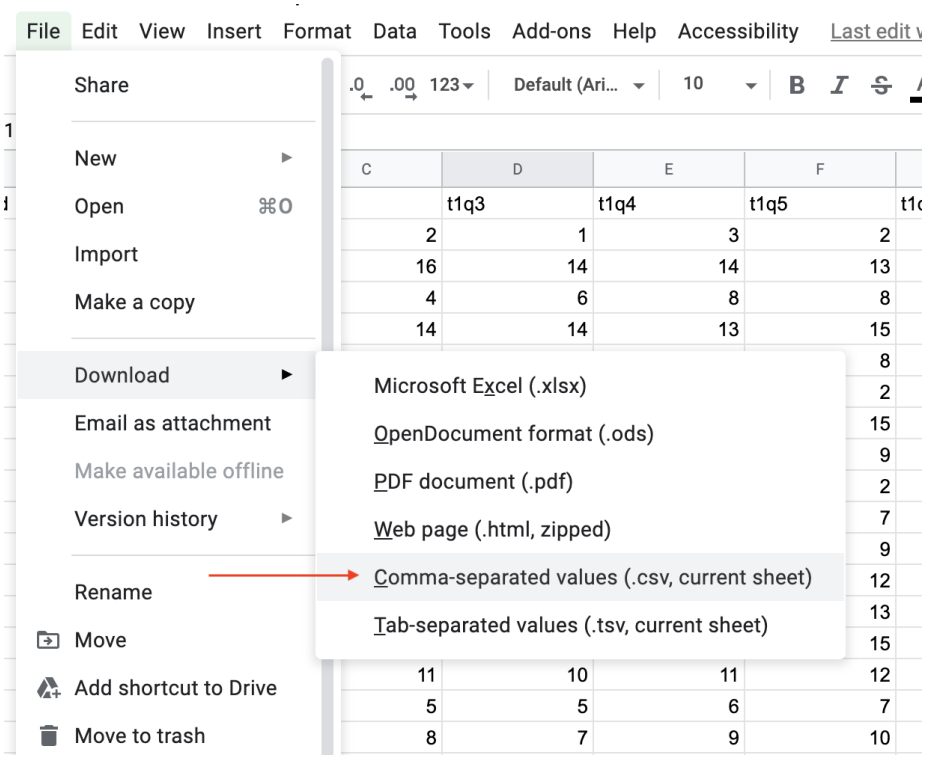
I	J	K	
t1q8	cmaq	genanx	sema
2	2	13	
13		12	
7		16	
11		8	
4		18	
2		15	
13		22	
12		7	
6		6	

	I	J	K	
	t1q8	cmaq	genanx	sema
4	2	2	13	
14	13	13.875	12	
3	7	5.625	16	
10	11	12.5	8	
5	4	5.875	18	
1	2	1.375	15	
12	13	12.875	22	
12	12	10	7	
6	6		6	
9	9		10	
8	8		12	
11	13		11	

When the cmaq column is complete (i.e. you have calculated the average score for each participant), save the data as a .csv file.

To do this in google sheets, select in the top bar menu:

FILE -> DOWNLOAD -> COMMA-SEPERATED VALUES (.CSV)



Again, the process is very similar in other spreadsheet programs.

Next import the data into your RStudio project and assign the dataset to an object called cmaqpilot. To assign the data to an object, you can use the point and click (GUI) method or you could use the following code:

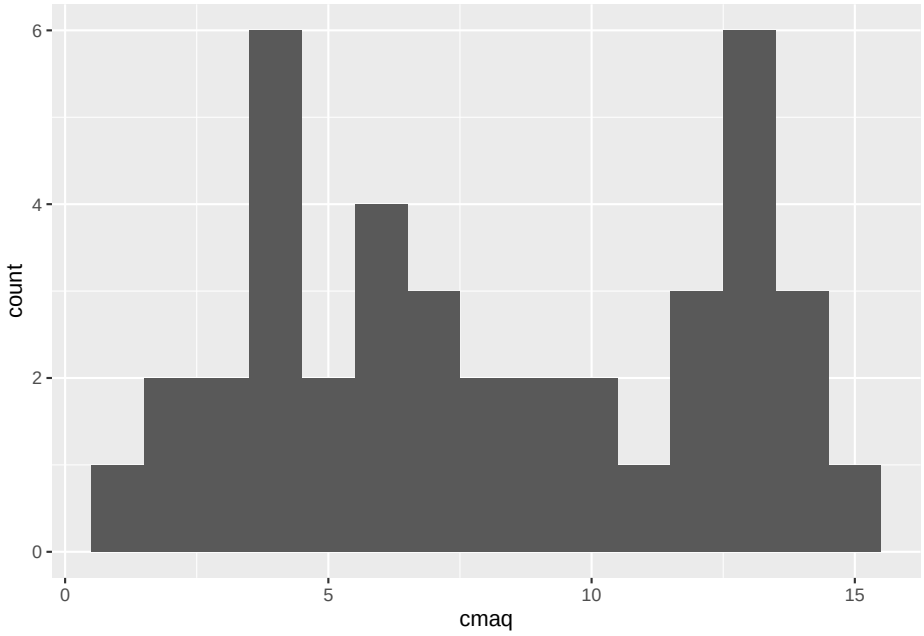
```
library(readr)
cmaqpilot <- read_csv("measurementma.csv")
```

Load the tidyverse and psych packages (if they are not loaded already):

```
library(tidyverse)
library(psych)
```

And then let's first create of histogram of the CMAQ scores:

```
ggplot(cmaqpilot, aes(x=cmaq)) +
  geom_histogram(binwidth=1)
```



The figure shows that all scores are within the range of possible scores, suggesting no errors occurred during data entry or when you calculated the average scores. The shape is slightly bi-modal, suggesting students are more likely to feel high or low levels of math anxiety, than to feel moderate amounts of math anxiety.

7.0.2 Reliability

The first step in establishing construct validity is to test the reliability of the measure. **Reliability** refers to consistency.

7.0.2.1 Internal reliability

For self-report measures, like the CMAQ, you need to measure internal reliability, which measures the extent to which people give consistent responses on every item of a survey. Researchers typically use Cronbach's alpha to test whether a measurement scale has internal reliability. Cronbach's alpha is essentially the average correlation of the correlations between each item of the scale (for a 3 item scale: the average of the correlations between item 1 and 2, item 1 and item 3, and item 2 and 3). This average is weighted by the average variance and the number of items, so it is not quite that simple - but it is the gist.

Like all correlations, Cronbach's alphas can technically range from -1 to 1. Higher Cronbach's alphas indicate better internal reliability (the correlations

between the scale items are higher). Cronbach's alphas of over .70 are considered acceptable in psychology.

I said technically above because negative Cronbach's alphas are almost unheard of. In the math anxiety example, that would mean that children reported feeling anxious for one item while not feeling anxious for another item. If all of the items are measuring the same thing (math anxiety), people should respond to them in a consistent matter.

In order to calculate the Cronbach's alpha in R you have to create a new object with only the items of the measurement scale.

```
cmaq <- cmaqpilot %>%
  select(t1q1, t1q2, t1q3, t1q4, t1q5, t1q6, t1q7, t1q8)
```

- This code tells R to select the variables listed in the select function from the cmaqpilot and save it as cmaq.

Then use the `alpha()` function, which is part of the psych package, to compute Cronbachs alpha of all if the variables in the cmaq object.

```
alpha(cmaq)
```

```
##
## Reliability analysis
## Call: alpha(x = cmaq)
##
##   raw_alpha std.alpha G6(smc) average_r S/N    ase mean  sd median_r
##       0.98      0.98    0.99      0.89  65 0.0036  8.3 4.1     0.89
##
##  lower alpha upper     95% confidence boundaries
## 0.98 0.98 0.99
##
##  Reliability if an item is dropped:
##      raw_alpha std.alpha G6(smc) average_r S/N alpha se   var.r med.r
## t1q1      0.98      0.98    0.98      0.88  53   0.0045 0.00122  0.88
## t1q2      0.98      0.98    0.98      0.88  52   0.0046 0.00098  0.88
## t1q3      0.98      0.98    0.98      0.89  59   0.0040 0.00099  0.89
## t1q4      0.98      0.98    0.99      0.89  56   0.0042 0.00158  0.88
## t1q5      0.98      0.98    0.99      0.89  57   0.0041 0.00126  0.89
## t1q6      0.98      0.98    0.98      0.89  55   0.0043 0.00156  0.88
## t1q7      0.98      0.98    0.99      0.90  62   0.0038 0.00117  0.90
## t1q8      0.98      0.98    0.99      0.90  63   0.0038 0.00123  0.90
##
##  Item statistics
```

```
##       n raw.r std.r r.cor r.drop mean  sd
## t1q1 40  0.97  0.97  0.97   0.96  7.7 4.8
## t1q2 40  0.98  0.98  0.98   0.97  8.5 4.6
## t1q3 40  0.94  0.94  0.94   0.92  8.4 4.2
## t1q4 40  0.95  0.95  0.95   0.94  8.7 4.3
## t1q5 40  0.95  0.95  0.94   0.94  8.2 4.7
## t1q6 40  0.96  0.96  0.96   0.95  8.0 4.3
## t1q7 40  0.93  0.93  0.92   0.91  8.4 3.9
## t1q8 40  0.92  0.93  0.91   0.90  8.3 4.0
```

In the output - focus on the raw alpha. In this example the Cronbach's alpha is .98, which is very high - indicating very good internal reliability (Cronbach's alpha with young kids are rarely this high - outing me for making up this data).

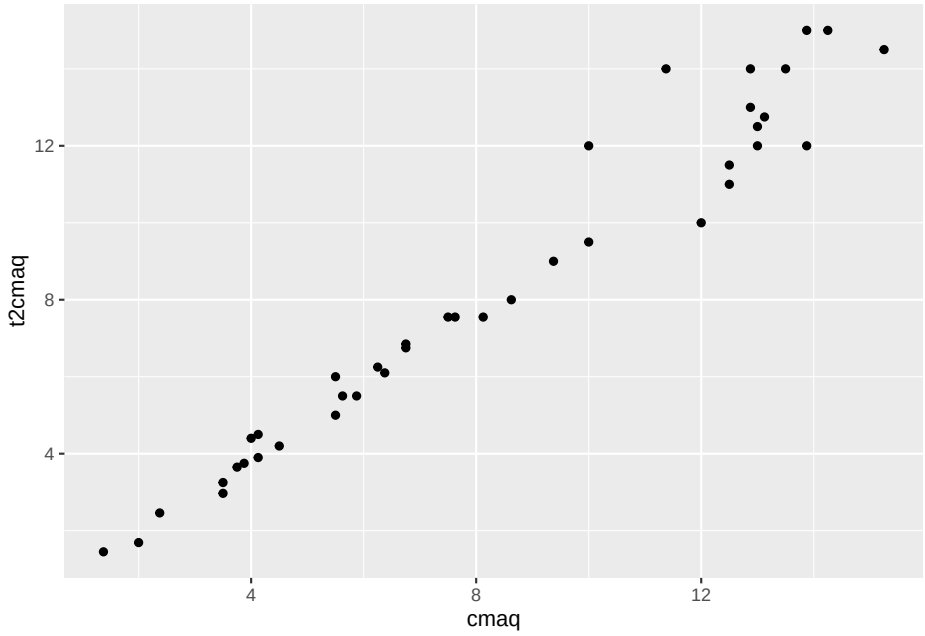
When Cronbach's alphas are less than .70, researchers have to revise and reconsider items. The purpose of the rest of the output is to get a sense of what Cronbach's alpha would be without an item. (Here is a good reference for more information about the rest of the alpha() function output if you are interested: https://rpubs.com/hauselin/reliabilityanalysis)

7.0.2.2 Test-retest reliability

Test-retest reliability refers to consistency of a measure over time. To test this we will use scatterplots and correlation coefficients.

Let's first create a scatterplot of the relation between the cmaq and the t2cmaq variable.

```
ggplot(cmaqpilot, aes(x=cmaq, y=t2cmaq)) +
  geom_point()
```



This scatterplot looks highly positive.

Let's next calculate a correlation coefficient between the cmaq and the t2cmaq variable. We will use the `corr()` function to calculate the confidence interval, effect size, and NHST (Null Hypothesis Significance Testing). The `corr.test()` function is part of the Psych package and the organization of the code below uses Tidyverse, so you should have both packages loaded.

Here is the code to compute the correlation coefficient between the cmaq and the t2cmaq variable:

```
cmaqpilot %>%
select(cmaq, t2cmaq) %>%
corr.test() %>%
print(short=FALSE)
```

- Add `method="spearman"` within the `corr.test()` parentheses for ranked data (For example: `corr.test(method = "spearman")`)
- The `short = FALSE` in the `print()` parentheses prints the confidence intervals
- Use `?corr.test` for more options

```
## Call:corr.test(x = .)
## Correlation matrix
##        cmaq t2cmaq
## cmaq   1.00   0.98
```

```
## t2cmaq 0.98   1.00
## Sample Size
## [1] 40
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##        cmaq t2cmaq
## cmaq      0      0
## t2cmaq    0      0
##
##  Confidence intervals based upon normal theory.  To get bootstrapped values, try cor.ci
##            raw.lower raw.r raw.upper raw.p lower.adj upper.adj
## cmaq-t2cmq      0.96  0.98      0.99     0      0.96      0.99
```

The first correlation matrix shows that the correlation between the cmaq and t2cmaq is .98.

The second matrix shows the NHST estimates the likelihood of getting results as extreme or more extreme given the null is true (i.e., given there is really no association between the variables). If this likelihood is sufficiently small (less than 5%), than we reject the null hypothesis and conclude that the association is more extreme than zero. When the probability value is listed as 0, you should report it as $p < .001$.

The last part of the output gives the confidence intervals around the correlation coefficient. The confidence interval provides an interval estimate of a parameter. Here the parameter is the true correlation between the two variables. In the present example, the correlation coefficient ($r = 0.98$) is a point estimate of the true association between the CMAQ at time 1 and 2. The confidence interval gives us an interval estimate of this association (it is between .96 to .99). Larger confidence intervals indicate more uncertainty about the true size of the association.

The scatterplot and correlation coefficient suggest that the CMAQ has test-retest reliability - they both show that the children responded to the CMAQ items consistently over time.

7.0.2.3 Interrater reliability

Interrater reliability refers to the consistency of coding ratings between different raters. We will have to use a different example to learn how to test interrater reliability because there is no observational measures in the math anxiety example.

The NICHD Early Child Care Research Network (1999) studied babies interactions with their mothers and child care providers over the first 3 years of life. They measured maternal sensitivity by observing mothers and their children during a semi-structured mother-child dyadic play procedure. The researchers measured maternal sensitivity by rating the amount of stimulation mothers

provided, responsiveness to non-distressed, intrusiveness, and positive regard during the play session.

Say you were responsible for validating the observational measure of maternal sensitivity before the NICHD Early Child Care Research began collecting their data.

You recruited 59 mother-child pairs to come to your lab. After you explained the purpose of the study and got consent, you recorded them during the semi-structured play procedure.

Then you and another researcher each watched the recordings (separately) and rated the mothers on the amount of stimulation mothers provided, their responsiveness when their child was not distressed, their intrusiveness, and their positive regard. You and the other researcher had a common codebook of behaviors to look for and were trained to recognize them.

The data is in matsen.csv

Open the data in RStudio.

```
library(readr)
matsen <- read_csv("matsen.csv")
```

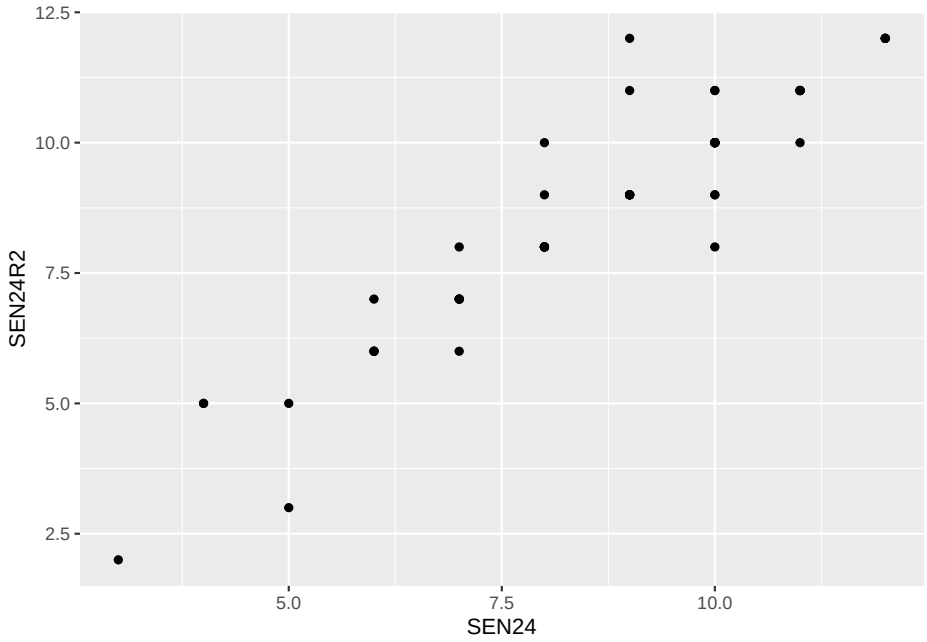
The first column is an arbitrary ID number. If you scroll down, you will see there are 59 mothers in total.

Next is your observational rating of each mother's sensitivity to her child during the semi-structured play session. The third column is the other researchers' observations.

Let's test the reliability of the observational measure of maternal sensitivity.

Let's first create a scatterplot:

```
library(tidyverse)
ggplot(matsen, aes(x=SEN24, y=SEN24R2)) +
  geom_point()
```



The scatterplot shows a strong positive relation between the two independent ratings of maternal sensitivity.

Next quantify the relation by computing a correlation coefficient.

```
matsen %>%
  select(SEN24, SEN24R2) %>%
  corr.test() %>%
  print(short=FALSE)
```

```
## Call:corr.test(x = .)
## Correlation matrix
##         SEN24 SEN24R2
## SEN24    1.00    0.94
## SEN24R2  0.94    1.00
## Sample Size
## [1] 59
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##         SEN24 SEN24R2
## SEN24       0       0
## SEN24R2     0       0
##
##  Confidence intervals based upon normal theory.  To get bootstrapped values, try cor.ci
##              raw.lower raw.r raw.upper raw.p lower.adj upper.adj
## SEN24-SEN24R       0.9  0.94      0.96     0       0.9      0.96
```

The results show that the correlation between the two raters is .94 with a 95% confidence interval of .90 to .96. This suggest strong agreement between raters.

7.0.3 Validity

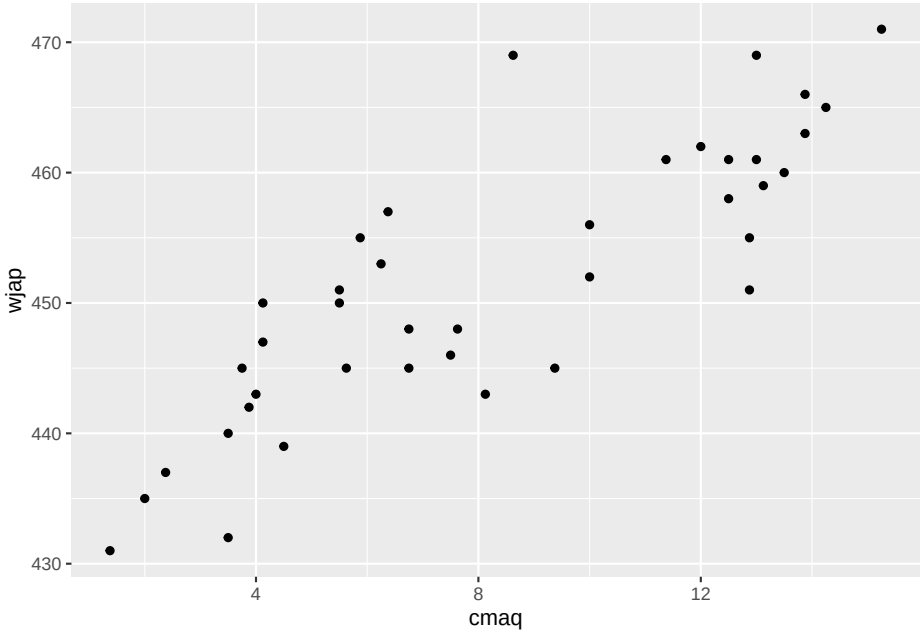
The next step in establishing the construct validity of the CMAQ is to establish that is has validity. **Validity** refers to accuracy. We will focus on the 3 empirical ways to assess validity here.

7.0.3.1 Criterion validity

Criterion validity refers to whether a measure is related to relevant behavioral outcomes.

In the current example, we will test whether the CMAQ is related to scores on the applied problems subscale of the Woodcock-Johnson III (WJ-III).

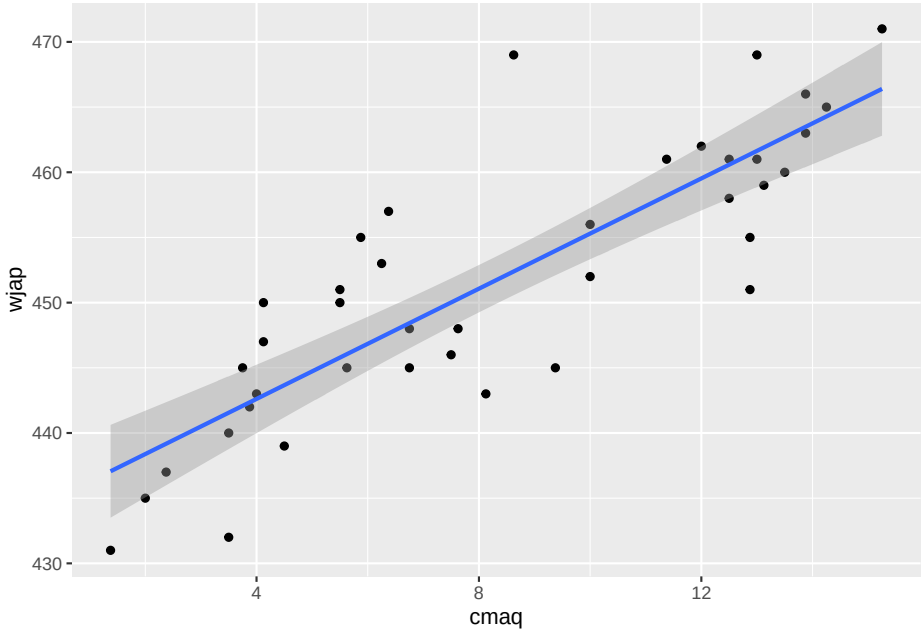
```
ggplot(cmaqpilot, aes(x=cmaq, y=wjap)) +
  geom_point()
```



This plot suggests that the CMAQ is strongly (and positively) correlated to scores on the applied problems subscale of the WJ-III, which is evidence for criterion validity of the CMAQ.

You could add the regression line to the scatterplot by adding `geom_smooth(method='lm')` to your code.

```
ggplot(cmaqpilot, aes(x=cmaq, y=wjap)) +
  geom_point() +
  geom_smooth(method='lm')
```



Some think that it is easier to see that the data points are close to the regression line. It also allows you to see the slope of the line - steeper lines indicate stronger relations.

Next calculate the correlation coefficient:

```
cmaqpilot %>%
  select(cmaq, wjap) %>%
  corr.test() %>%
  print(short=FALSE)
```

```
## Call:corr.test(x = .)
## Correlation matrix
##      cmaq wjap
## cmaq 1.00 0.84
## wjap 0.84 1.00
## Sample Size
## [1] 40
```

```
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##      cmaq wjap
## cmaq    0    0
## wjap    0    0
##
##  Confidence intervals based upon normal theory.  To get bootstrapped values, try co
##           raw.lower raw.r raw.upper raw.p lower.adj upper.adj
## cmaq-wjap      0.72  0.84      0.91     0      0.72      0.91
```

CMAQ scores were positively related to students' applied problems WJ-III scores (r = .84, p < .001, CI.95 = .72 to .91).

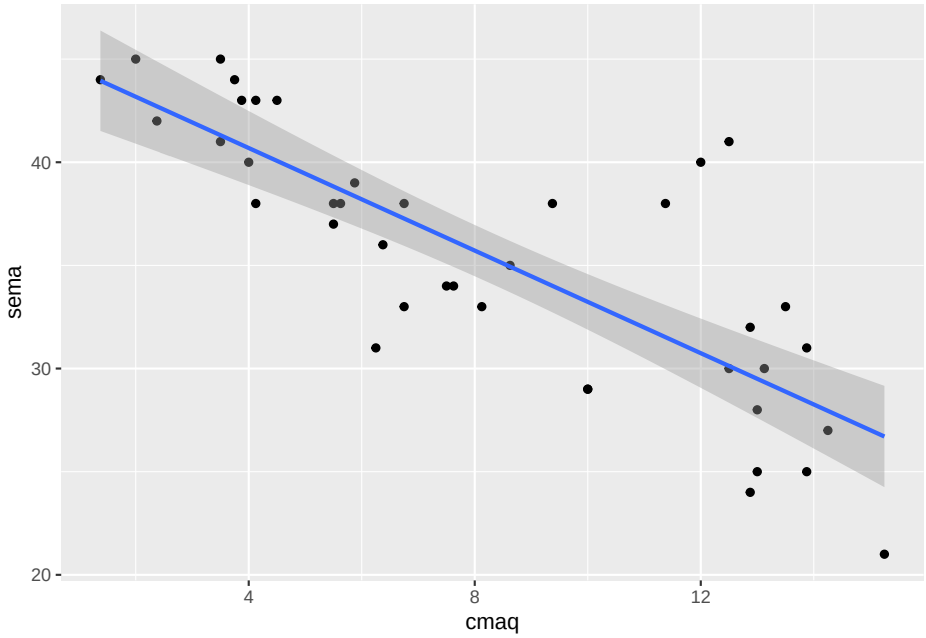
The scatterplot and correlation show that the CMAQ is highly related to a behavioral measure of math ability, the applied problems WJ-III scores. [I guess math ability is not the same as math anxiety - despite evidence that they are strongly related. Perhaps a neuro-based variable would have been better here?]

7.0.3.2 Convergent and Discriminant Validity

Convergent and discriminant validity are often considered together. Convergent validity is whether a measure is related to similar measures. Discriminant validity is whether a measure is not related to dissimilar measures.

In the current example, we will test whether the CMAQ is related to the SEMA, which is another measure of math anxiety. Remember that higher scores on the SEMA indicates high levels of math anxiety. While low scores on the CMAQ indicate high levels of math anxiety. So a negative relation here would indicate convergent validity.

```
ggplot(cmaqpilot, aes(x=cmaq, y=sema)) +
  geom_point() +
  geom_smooth(method='lm')
```



The figure shows that the CMAQ is negatively correlated to the SEMA.

Next calculate the correlation coefficient:

```
cmaqpilot %>%
  select(cmaq, sema) %>%
  corr.test() %>%
  print(short=FALSE)
```

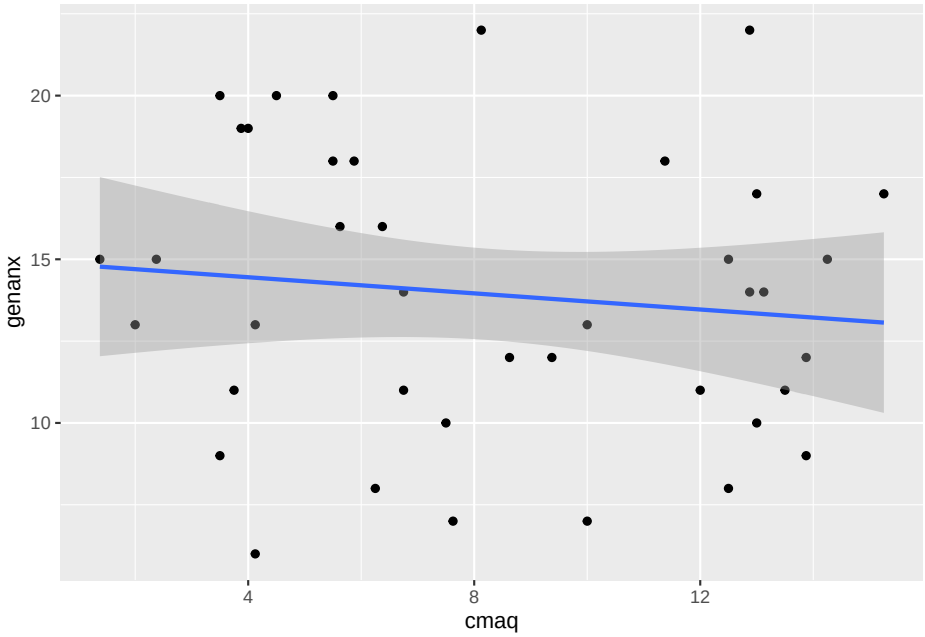
```
## Call:corr.test(x = .)
## Correlation matrix
##      cmaq sema
## cmaq  1.0 -0.8
## sema -0.8  1.0
## Sample Size
## [1] 40
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##      cmaq sema
## cmaq    0    0
## sema    0    0
##
##  Confidence intervals based upon normal theory.  To get bootstrapped values, try cor.ci
##           raw.lower raw.r raw.upper raw.p lower.adj upper.adj
## cmaq-sema     -0.89  -0.8     -0.65     0     -0.89     -0.65
```

CMAQ scores were negatively related to students' SEMA scores (r = -.80, p

< .001, CI.95 = -.89 to -.65). That is, children reported equal levels of math anxiety on the CMAQ and SEMA.

In the current example, we will test discriminant validity by testing whether the CMAQ is related to general anxiety. Here a zero relation would indicate convergent validity.

```
ggplot(cmaqpilot, aes(x=cmaq, y=genanx)) +
  geom_point() +
  geom_smooth(method='lm')
```



The figure shows a zero correlation between CMAQ and SEMA.

Next calculate the correlation coefficient:

```
cmaqpilot %>%
  select(cmaq, genanx) %>%
  corr.test() %>%
  print(short=FALSE)
```

```
## Call:corr.test(x = .)
## Correlation matrix
##         cmaq genanx
## cmaq    1.00  -0.12
## genanx -0.12   1.00
```

```
## Sample Size
## [1] 40
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##        cmaq genanx
## cmaq   0.00   0.47
## genanx 0.47   0.00
##
##  Confidence intervals based upon normal theory.  To get bootstrapped values, try cor.ci
##            raw.lower raw.r raw.upper raw.p lower.adj upper.adj
## cmaq-gennx     -0.41 -0.12       0.2  0.47     -0.41       0.2
```

CMAQ scores were not related to students' general anxiety scores (r = -.12, p = . 47, CI.95 = -.41 to .20). Note that the confidence interval here includes zero, which is consistent with NHST because both are saying that zero is a likely correlation between the variables.

Taken together, the CMAQ seems to have convergent and discriminant validity because it is highly related to another measure of math anxiety and it is not related to general anxiety.

Chapter 8

Basic Data Transformations

NOTE: please create a new script for this chapter and the interrater relatiablity seciton called week 4

For this section we will use a dataset from SPSS for Research Methods by Wilson-Doenges, which comes with our Morling text. Here is the survey that Wilson-Doenges distributed to 45 students.

You can find the data on D2L called `'wilson.csv'`.

Please download it and take a few minutes to look over the survey (open the link above in the word 'here') and study how Wilson-Doenges entered in her data.

The first column is an arbitrary ID number assigned to each student to ensure anonymity. The next 4 columns correspond with the first four questions of the survey.

The second part of the survey measures students' positive opinions about a research methods class. Wilson calls it the positive opinions about research methods scale (PORMS). In the data file, these are item1, item2, item3, item4, and item5.

The last two questions of the survey ask students to report their motivation to achieve and their GPA (the last two columns).

First load the readr and tidyverse packages (if they are not loaded already):

```
library(readr)
library(tidyverse)
```

Then assign the data to an object using the following code (or you can use the GUI method):

```
wilson <- read_csv("wilson.csv")
```

While you were looking at the survey, you may have noticed that items 2 and 4 of the PORMS are negatively worded; While items 1, 3, and 5 are positively worded. This means that strongly agree (i.e. the number 5) indicates that students have a negative opinion of research methods classes for items 1 and 4 and that they have a positive opinion of the class for items 1, 3, and 5.

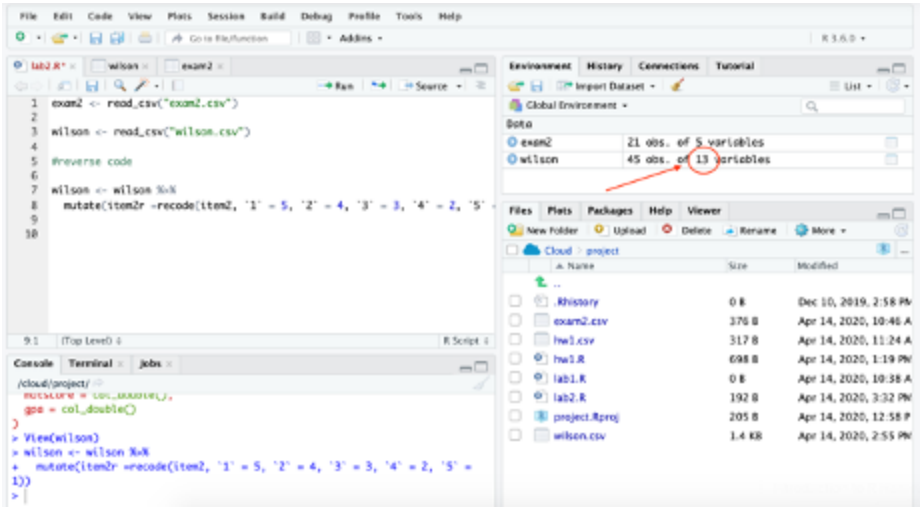
We need all of the items to go in the same direction. So, we need to **reverse code** items 2 and 4 so that higher scores reflect more positive opinions. To reverse code, we will use the **mutate()** and **recode()** functions of the dplyr package (that is part of tidyverse), which adds new variables or changes existing ones. * mutate() is used to add variables (or columns) to a dataset. * recode() is best used inside a mutate (). Recode takes the form of old_value = new_value.

The command to reverse code item 2 is:

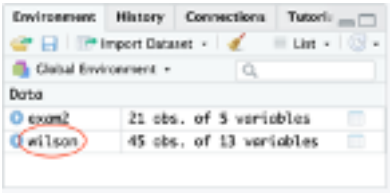
```
wilson <- wilson %>%
   mutate(item2r =recode(item2, `1` = 5, `2` = 4, `3` = 3, `4` = 2, `5` = 1))
```

- `item2r` will be the name of the new variable.
- `item2` is the item that is being recoded.
- Next is the list of the old and new variables
  - On the left is the old variable and it must be in back ticks (‘) when it is a number
  - Note that the back tick is not the same as a comma ('). The back tick is on the same key as ~ (while the comma is on the same key as ")
  - String (AKA text) variables should be in quotes (") instead of back ticks
  - On the right is the new value
- The `wilson <-` part of the command saves the variable you created with the rest of the code
  - Here we are saving over the original dataset.
  - Some people prefer to create a new dataset. For example `wilsonr <-` would create a new object called wilsonr and the wilson data would not change.
  - Without this part of the code, your new variable will not be saved.

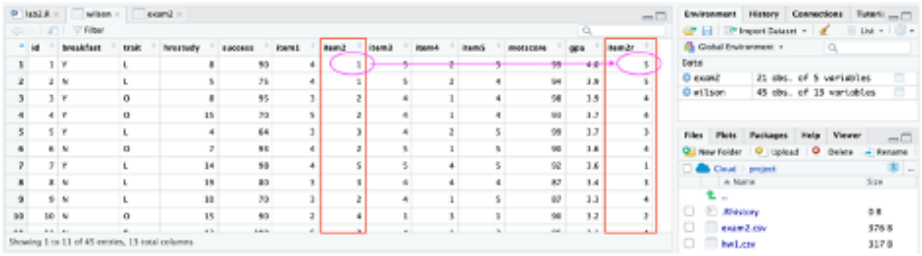
After you run the code, there should be 13 variables in the wilson dataset (there was 12 orginally).



Click on the word 'wilson' in the environment panel to view the data.



The reverse coded item 2 variable (item2r) that we just created will be in the last column.



(You can expand the data view by dragging the center median between the dataview and the environment to the right.)

You can see that the first student rated the second item as a 1 and it is now a 5 in the reversed coded variable.

Next create a new item for item 4. Here is the code:

```
wilson <- wilson %>%
mutate(item4r =recode(item4, `1` = 5, `2` = 4, `3` = 3, `4` = 2, `5` = 1))
```

You should now have 14 variables in the wilson dataset.

Next let's create a summary score for the PORMS measure. We will use the mutate() function to do this using this code:

```
wilson <- wilson %>%
mutate(porms = item1 + item2r + item3 + item4r + item5)
```

- porms is the name of the new column
- On the right of the equal sign is how the new variable is defined.

Your wilson dataset should now have 15 variables.

(I created a sum score here because that is what Wilson-Doenges did. I think an average score would work here as well.)

Chapter 9

Bivariate correlational research

9.1 Association claim with two quantitative variables

9.1.0.1 Open data

9.1.0.2 Get to know data

9.1.0.3 Test assumptions

9.1.0.4 Compute CI, effect size, and NHST

9.1.0.5 Write up results

Chapter 10

Multivariate correlational research

Chapter 11

Simple experiment

11.1 Independent group design

11.1.1 Two groups

11.1.2 More than two groups

11.2 Dependent group design

11.2.1 Two groups

11.2.2 More than two groups

Chapter 12

Experiments with more than one IV

12.1 Independent-group factorial design

12.2 Within-group factorial design

12.3 Mixed factorial design

Chapter 13

Final Words